ComponentOne

Entity Framework DataSource

ComponentOne, a division of GrapeCity

201 South Highland Avenue, Third Floor

Pittsburgh, PA 15206 • USA

Internet: **info@ComponentOne.com**

Web site: **http://www.componentone.com**

Sales

E-mail: sales@componentone.com

Telephone: 1.800.858.2739 or 1.412.681.4343 (Pittsburgh, PA USA Office)

This manual was produced using ComponentOne Doc-To-Help™.

Table of Contents

ComponentOne Entity Framework DataSource Overview

ComponentOne Entity Framework DataSource (EF DataSource) adds ease-of-use and performance enhancements to the Entity Framework and RIA Services. It improves and simplifies data binding with these frameworks by solving common problems related to loading, paging, filtering and saving data. It also provides performance enhancements such as fast loading and transparent scrolling over large datasets with Virtual Mode.

Help with the ComponentOne Studios

Getting Started

For information on installing any of the ComponentOne Studios, licensing, technical support, namespaces and creating a project with the control, please visit:

- Getting Started with Studio for WinForms
- Getting Started with Studio for WPF
- Getting Started with Studio for Silverlight

What's New

For a list of the latest features added to the ComponentOne Studios, please visit:

- What's New in Studio for WinForms
- What's New in Studio for WPF
- What's New in Studio for Silverlight

Key Features

The following are some of the main features of **ComponentOne Entity Framework DataSource** (**EF DataSource**) that you may find useful:

- **Cross Platform Support**
  EF DataSource includes a **C1DataSource** component which allows you to combine multiple client view sources using Entity Framework or RIA Services. **C1DataSource** is supported in Silverlight 4, WPF and even WinForms (.NET 4.0).

- **Smart Client-side Caching**
  The key to most of **EF DataSource**'s features is its built-in client-side data cache. **EF DataSource** maintains a cache of entities on the client. When new queries are executed, it does not necessarily go to the server. It checks the client-side cache first and will not go to the server if it can find the result in the cache. This significantly improves performance and speed because it minimizes the number of trips to and from the server.

- **Context Management**
  With **EF DataSource** you can use a single data context for your entire application. This allows you to forget the troubles of programming against multiple contexts across multiple views. Without **EF DataSource** you might have multiple contexts which can be very difficult to write code when you need to use entities from different contexts together. This feature is made possible by the smart client-side cache.

- **Memory Management**
  EF DataSource is optimized for both performance and memory consumption. It manages memory resources for you releasing old entity objects in the cache when necessary to prevent memory leaks. In doing so, it fully preserves consistency by maintaining required entities and entities modified by the user.

- **Live Views with LiveLinq**
  LINQ is the perfect tool for transforming raw data into custom views. **EF DataSource** makes LINQ even more powerful by making LINQ query statements live. **EF DataSource** includes ComponentOne LiveLinq, an extension library which augments the functionality of LINQ to speed up queries and provide live views. With LiveLinq you can shape your view however you want using LINQ operators without losing full updatability and bindability. "Bindability" means that your views are not just static snapshots of their source data. They are "live" and automatically reflect changes in the data. With LiveLinq your query results are kept up-to-date without re-populating every time changes in your data occur.

- **Virtual Mode for Large DataSets**
  Handle infinitely large datasets with **EF DataSource**'s Virtual Mode. Virtual Mode allows you to navigate through large datasets asynchronously. It works like paging on the data layer but the user can scroll through the data as if all rows were on the client. As the user scrolls, chunks of data are retrieved from the source page by page and disposed of as necessary. You can use Virtual Mode with any GUI control, such as a **DataGrid**, **C1FlexGrid**, or any control you want. Data can also be modified. This feature is transparent to the developer; you can turn on Virtual Mode with one simple property setting.

- **Paging with Data Modification**
  For applications that prefer a paging interface, **EF DataSource** also supports paging without any limitations on data modification. That means users can make changes on multiple pages in one session before having to push the changes back to the database. This is a substantial improvement over other paging implementations such as with the Microsoft RIA Services DomainDataSource. Paging with **C1DataSource** is also supported in WinForms and WPF too where paging is not provided.

- **Server-side Filtering**
  Data brought from the server to the client usually needs to be filtered, or restricted in some way, to avoid moving large amounts of data over the wire and heaping it on the client. **EF DataSource** makes this common task very simple. Instead of doing it manually in code, you can specify server-side filters as simple property settings on **C1DataSource**.

- **Client-Side Transactions EF DataSource** gives developers a simple and powerful mechanism for rolling back (cancelling) and accepting changes on the client without

involving the server. **EF DataSource** makes it easy to implement **Cancel**/**Undo** and **OK**/**Accept** buttons anywhere, even in nested (child) dialog boxes and forms, modal and modeless.

- **Saving Modified Data EF DataSource**'s smart client caching makes it easy to write code that saves modified data back to the server. With just one line of code and one trip to the server, you can save multiple entities. EF DataSource does all of the heavy lifting; it maintains changed entities in the cache, and ensures the cache is always consistent, while also optimizing memory usage and performance.

- **Code Freedom EF DataSource** allows you to set up your data sources directly on the designer surface, with easy-to-use property dialogs and very little code to write. Configure the **C1DataSource** control and apply server-side filter, sort and group descriptors quickly at design-time. Of course, if you prefer to do everything in code you have the freedom of doing so using the rich data class libraries.

- **Simplifies MVVM EF DataSource** can simplify programming with the widely-adopted Model-View-ViewModel pattern known as MVVM. You have to write a lot more code to develop MVVM applications because of the additional code layer, the ViewModel, and the synchronization between the Model and ViewModel data members. With EF DataSource you can use live views as your ViewModel and not have to worry about writing any synchronization code. Live views are synchronized automatically with their sources and are much easier to create. You can use **EF DataSource** with any MVVM framework.

ComponentOne Entity Framework DataSource Quick Start Guides

The following quick start guides will show you how to get started with **ComponentOne Entity Framework DataSource** in the WinForms, WPF, and Silverlight platforms. Each guide walks you through the steps of adding a data source, connecting it with data, and adding a grid to display the data. For a more detailed explanation on binding, see the **Simple Binding** topics for each of the platforms.

WinForms Quick Start

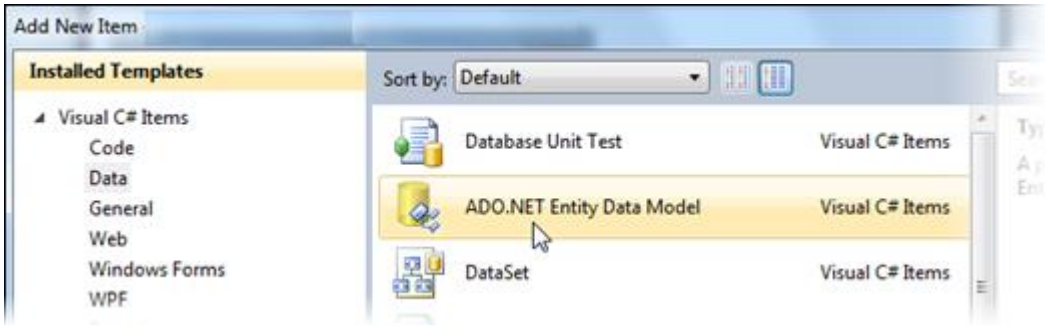
Getting started with **ComponentOne Entity Framework DataSource** only takes a few simple steps. You need to add a data source to your project, connect it to a C1DataSource component, and add a grid for displaying the data. This quick start shows the most basic steps for binding. For a more detailed explanation of the process, see the Simple Binding topic in this documentation.

This quick start tutorial uses the Northwind database that is installed with the product in the **Studio for WinForms\C1DataSource\Data** folder within *C:\Users\<username>\Documents\ComponentOne Samples* (Windows 7/Vista) or *C:\Documents and Settings\<username>\My Documents\ComponentOne Samples* (Windows XP).

Step 1 of 4: Adding a Data Source

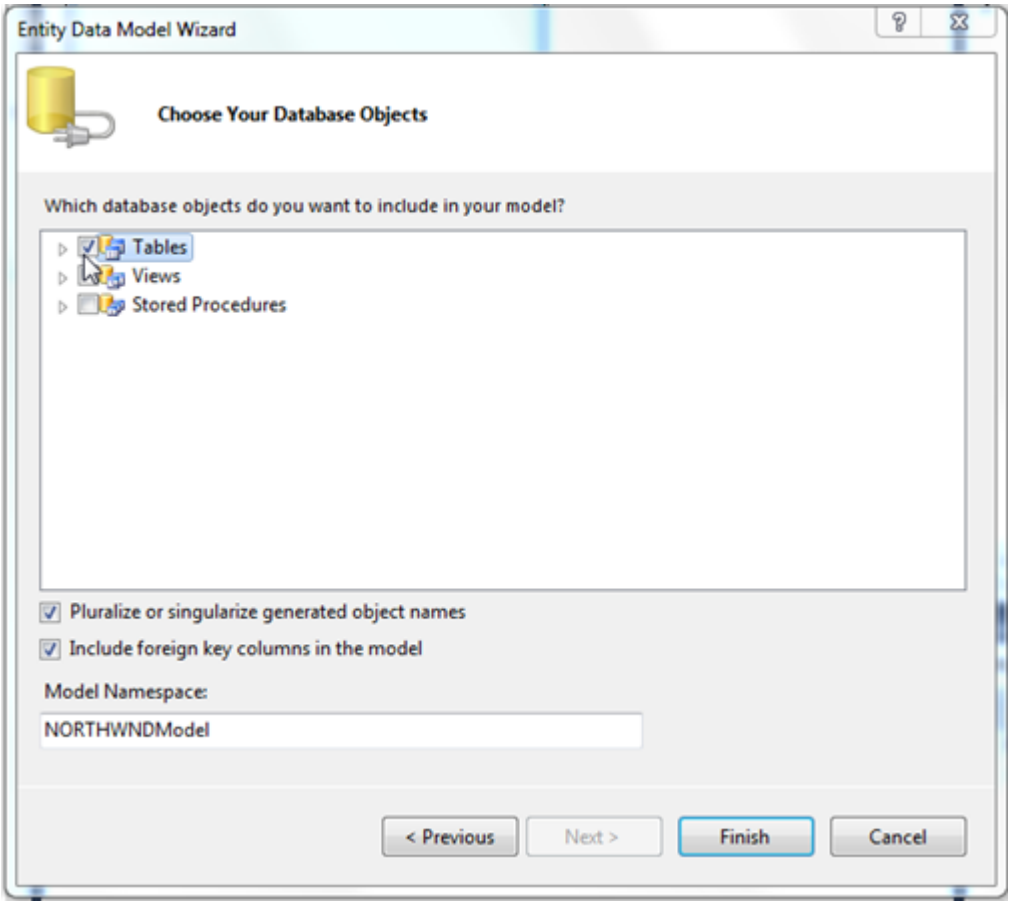
Begin by creating a new Windows Forms project in Visual Studio. Then you will add a connection to the Northwind database.

1. In Visual Studio, choose **File | New | Project.**
2. Select the **Windows Forms Application** template and click **OK**.
   Next, add an Entity Data Model based on the Northwind database. This model will provide the data for the entire application.
3. In the **Solution Explorer**, right-click the project name and select **Add | New Item**.
4. Choose the **ADO.NET Entity Data Model** item and then click **Add**.



5. In the **Entity Data Model Wizard**, select **Generate from database** to generate the model from the Northwind database, and then click **Next**.
6. Click the **New Connection** button.
7. In the **Choose Data Source** dialog box, select **Microsoft SQL Server Database File** and click **Continue**.
8. Browse to find the NORTHWND.MDF, select it, and click **Open**. The NORTHWND.MDF is installed with the product in the **Studio for WinForms\C1DataSource \Data** folder within *C:\Users\<username>\Documents\ComponentOne Samples* (Windows 7/Vista) or *C:\Documents and Settings\<username>\My Documents\ComponentOne Samples* (Windows XP).

9. Click **OK** and then click **Next**.
10. In the **Choose Your Database Objects** window, select **Tables** and click **Finish**.

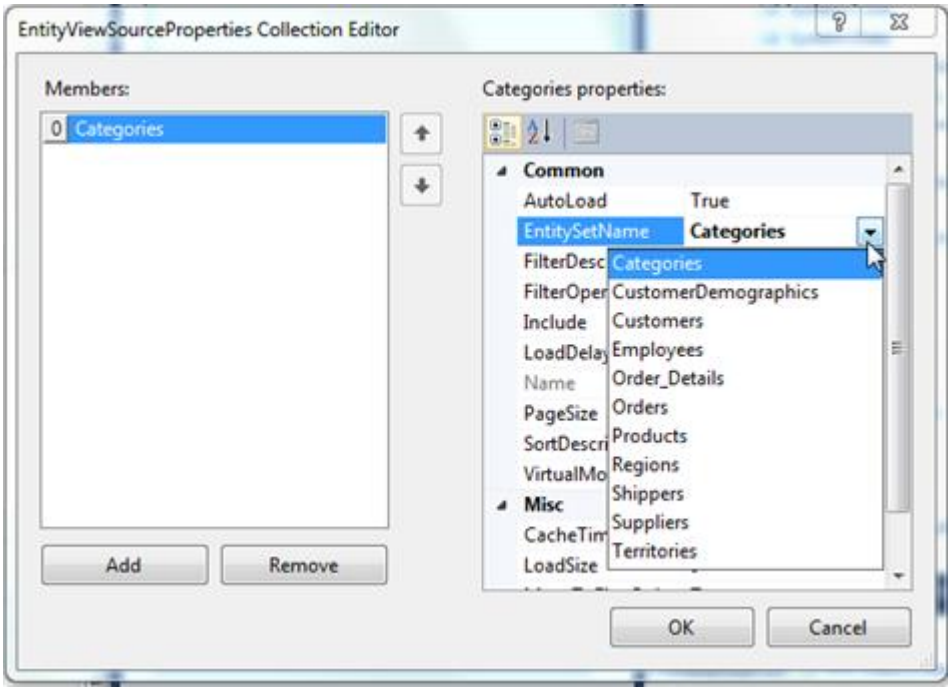


Now build the project so the new Entity Data Model classes are generated and become available throughout the project. Once they are available, you will be able to connect the data to the C1DataSource component.

Step 2 of 4: Connecting the Data to C1DataSource

In this step, you'll add a C1DataSource component to the form and connect it to the *Categories* table of the data source.

1. Drag a C1DataSource component from the Toolbox onto the form. This is a non-visual component, so it will appear on the tray below the form area rather than on the form itself.
2. In the **Properties** window, set the C1DataSource's **ObjectContextType** property to the available item in the drop-down list. It should be something similar to *AppName.NORTHWNDEntities*.
3. Click the **ellipsis** button next to the **ViewSourceCollection** property to open the **EntityViewSourceProperties Collection Editor**.
4. Click **Add** and set the **EntitySetName** property to *Categories*.



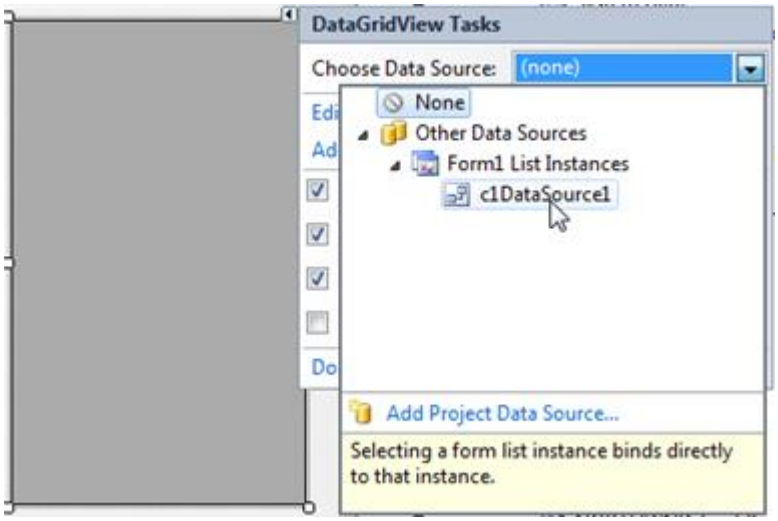
5. Click **OK** to close the editor.

   With the database connected to C1DataSource, all you need to do is add a grid to display the data.

Step 3 of 4: Adding a Grid

In this step you will add a grid that will be used to display the data in the *Categories* table of the Northwind database. You can use **C1FlexGrid** or any grid you are familiar with, but in this example, we'll use a **DataGridView** control.

1. Drag a **DataGridView** control from the **Toolbox** onto your form.
2. Click the **DataGridView**'s smart tag and set the **Data Source** to the **c1DataSource1** control we added to the form.



3. In the Visual Studio Properties window, set the **DataMember** property to *Categories*. The grid will automatically generate columns for all the fields in the *Categories* type.

   Now simply run the project to view the grid!

Step 4 of 4: Running the Project

Press **F5** to run the project. The data from the *Categories* table of the Northwind database is displayed.

WPF Quick Start

Getting started with **ComponentOne Entity Framework DataSource** only takes a few simple steps. You need to add a data source to your project, connect it to a C1DataSource component, and add a grid for displaying the data. This quick start shows the most basic steps for binding. For a more detailed explanation of the process, see the Simple Binding topic in this documentation.

This quick start tutorial uses the Northwind database that is installed with the product in the **Studio for WPF\C1.WPF.DataSource\Data** folder within *C:\Users\<username>\Documents\ComponentOne Samples* (Windows 7/Vista) or *C:\Documents and Settings\<username>\My Documents\ComponentOne Samples* (Windows XP).

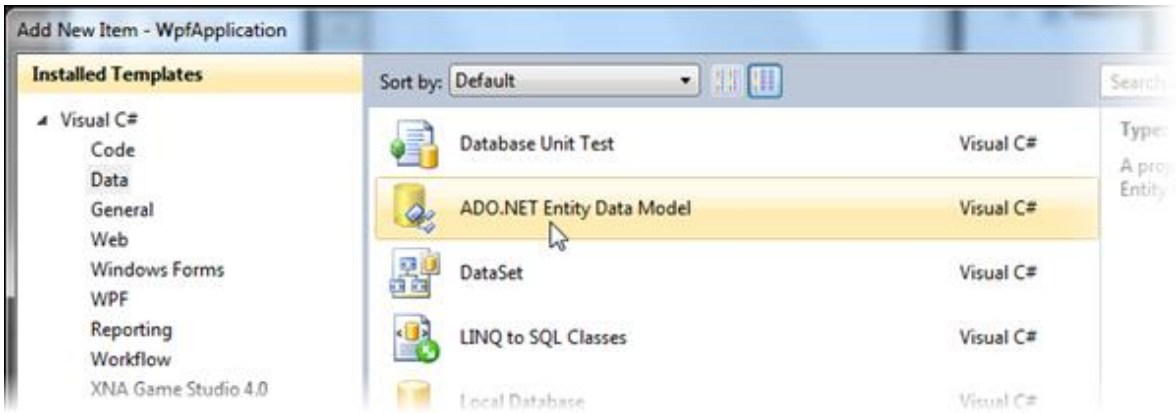
Step 1 of 4: Adding a Data Source

Begin by creating a new WPF project in Visual Studio. Then you will add a connection to the Northwind database.

1. In Visual Studio, choose **File | New | Project.**
2. Select the **WPF Application** template and click **OK**.

   Next, add an Entity Data Model based on the Northwind database. This model will provide the data for the entire application.

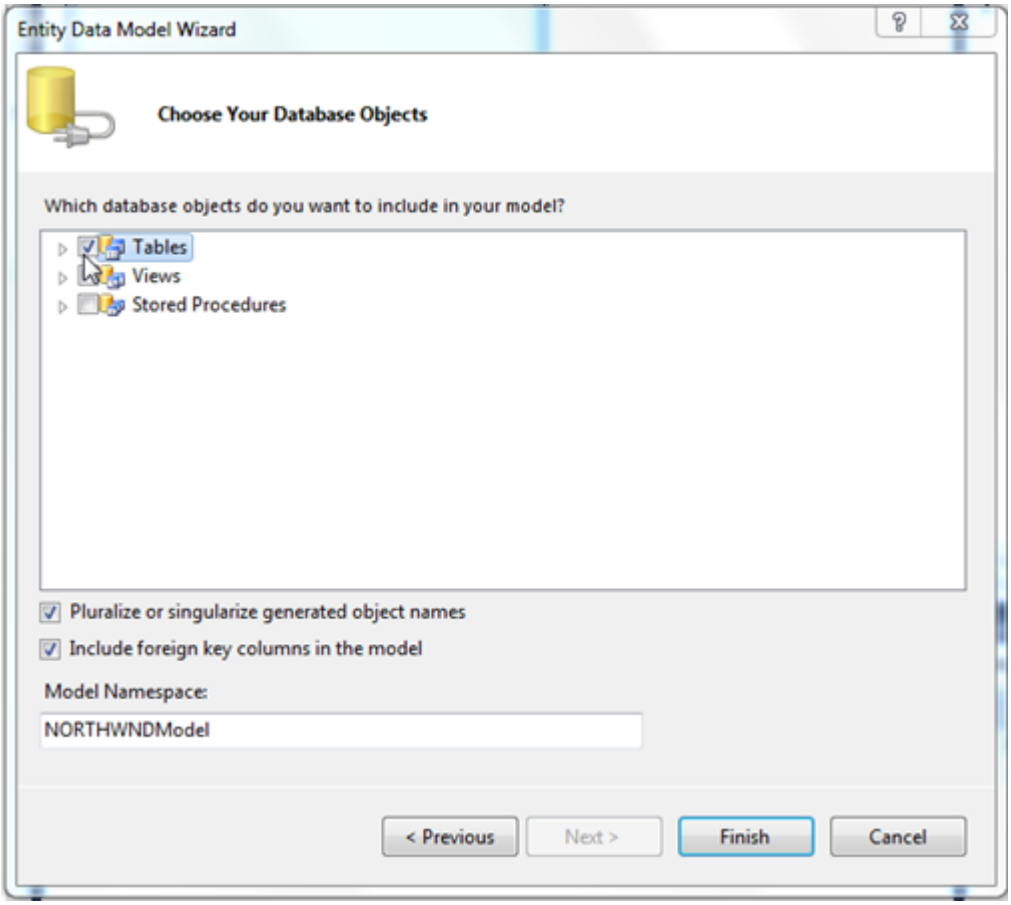
3. In the **Solution Explorer**, right-click the project name and select **Add | New Item**.
4. Choose the **ADO.NET Entity Data Model** item and then click **Add**.



5. In the **Entity Data Model Wizard**, select **Generate from database** to generate the model from the Northwind database, and then click **Next**.
6. Click the **New Connection** button.

7. In the **Choose Data Source** dialog box, select **Microsoft SQL Server Database File** and click **Continue**.
8. Browse to find the NORTHWND.MDF, select it, and click **Open**. The NORTHWND.MDF is installed with the product in the **Studio for WPF\C1.WPF.DataSource\Data** folder within *C:\Users\<username>\Documents\ComponentOne Samples* (Windows 7/Vista) or *C:\Documents and Settings\<username>\My Documents\ComponentOne Samples* (Windows XP).

9. Click **OK** and then click **Next**.
10. In the **Choose Your Database Objects** window, select **Tables** and click **Finish**.

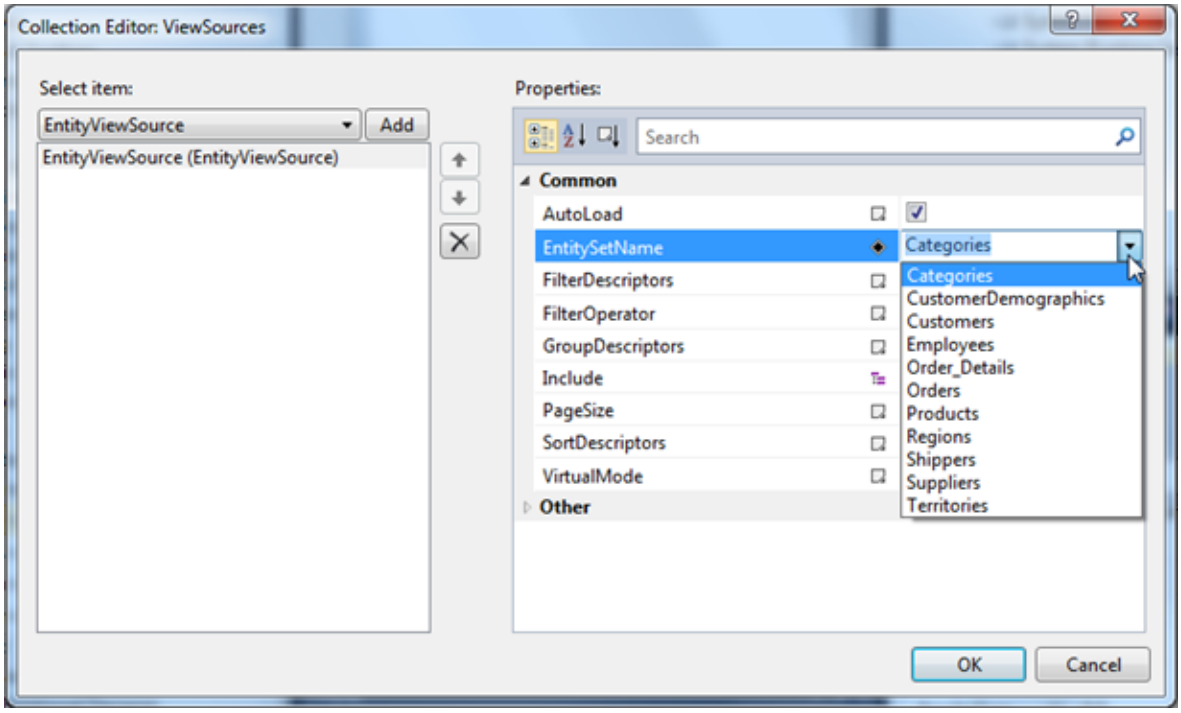


Now build the project so the new Entity Data Model classes are generated and become available throughout the project. Once they are available, you will be able to connect the data to the C1DataSource component.

Step 2 of 4: Connecting the Data to C1DataSource

In this step, you'll add a C1DataSource component to the window (or page) and connect it to the *Categories* table of the data source.

1. Drag a C1DataSource component from the Toolbox onto the window and name it **c1DataSource1**. This is a non-visual component, so it can be placed anywhere within the window's inner content.
2. In the **Properties** window, set the C1DataSource's **ObjectContextType** property to the available item in the drop-down list. It should be something similar to *AppName.NORTHWINDEntities*.
3. Click the **ellipsis** button next to the **ViewSources** property to open the **ViewSources Collection Editor**.
4. Click **Add** and set the **EntitySetName** property to *Categories*.



5. Click **OK** to close the editor.

   With the database connected to C1DataSource, all you need to do is add a grid to display the data.

Step 3 of 4: Adding a Grid

In this step you will add a grid that will be used to display the data in the *Categories* table of the Northwind database. You can use **C1FlexGrid** or any grid you are familiar with, but in this example, we'll use a **DataGrid** control.

1. Drag a **DataGrid** control from the **Toolbox** onto your window.
2. Specify a binding for the **DataGrid**'s **ItemsSource** property in XAML:
3.
4. `ItemsSource="{Binding [Categories], ElementName=c1DataSource1}"`
5. Set the **DataGrid**'s **AutoGenerateColumns** property to True; otherwise, the grid will not have any columns at run time.

   Now simply run the project to view the grid!

Step 4 of 4: Running the Project

Press **F5** to run the project. The data from the *Categories* table of the Northwind database is displayed.

CategoryID	CategoryName	Description
1	Beverages	Soft drinks, coffees, teas, beers, and ales
2	Condiments	Sweet and savory sauces, relishes, spreads, and se
3	Confections	Desserts, candies, and sweet breads
4	Dairy Products	Cheeses
5	Grains/Cereals	Breads, crackers, pasta, and cereal
6	Meat/Poultry	Prepared meats
7	Produce	Dried fruit and bean curd
8	Seafood	Seaweed and fish

Silverlight Quick Start

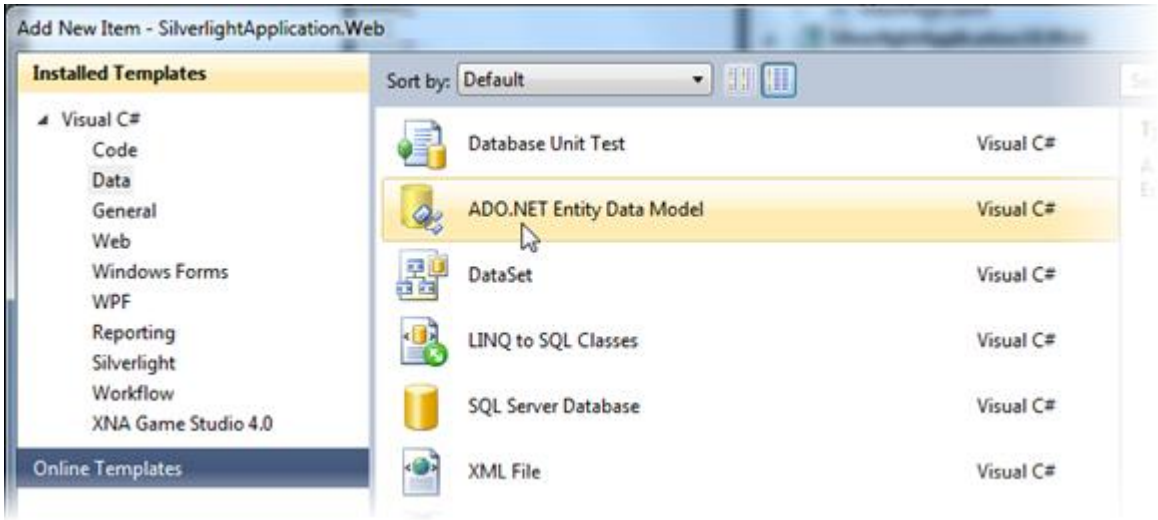
Getting started with **ComponentOne Entity Framework DataSource**only takes a few simple steps. You need to add a data source and a domain service class to your project, connect it to a C1DataSource component, and add a grid for displaying the data. This quick start shows the most basic steps for binding. For a more detailed explanation of the process, see the Simple Binding topic in this documentation.

This quick start tutorial uses the Northwind database that is installed with the product in the **Studio for Silverlight\C1.Silverlight.DataSource\Data** folder within *C:\Users\<username>\Documents\ComponentOne Samples* (Windows 7/Vista) or *C:\Documents and Settings\<username>\My Documents\ComponentOne Samples* (Windows XP).

Step 1 of 5: Adding a Data Source

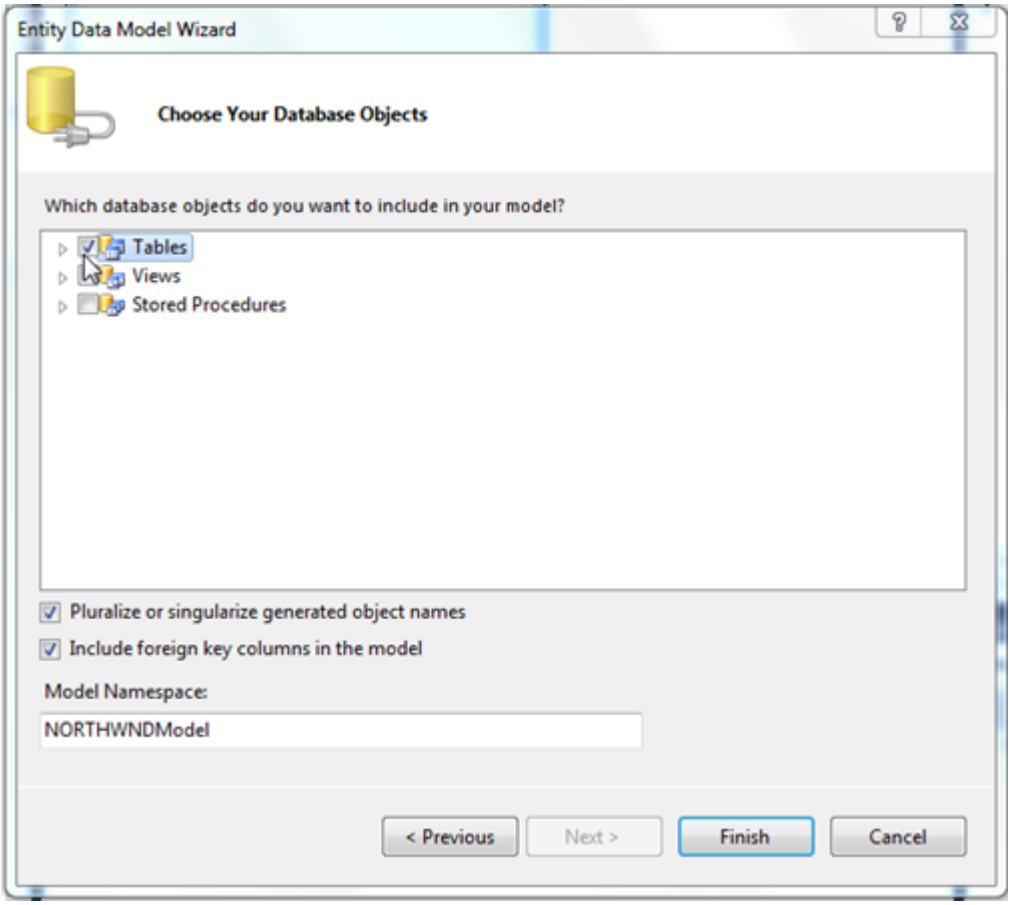
Begin by creating a new Silverlight project in Visual Studio. Then you will add a connection to the Northwind database.

1. In Visual Studio, choose **File | New | Project.**
2. Select the **Silverlight Application** template and click **OK**.
3. In the **New Silverlight Application** dialog box, select the **Enable WCF RIA Services** checkbox and click **OK**. The wizard will create two projects: one is a Silverlight project (client) and the other is a Web project (server). Next, add an Entity Data Model based on the Northwind database. This model will provide the data for the entire application.
4. In the **Solution Explorer**, right-click the Web project name and select **Add | New Item**.
5. Choose the **ADO.NET Entity Data Model** item and then click **Add**.



6. In the **Entity Data Model Wizard**, select **Generate from database** to generate the model from the Northwind database, and then click **Next**.
7. Click the **New Connection** button.
8. In the **Choose Data Source** dialog box, select **Microsoft SQL Server Database File** and click **Continue**.
9. Browse to find the NORTHWND.MDF, select it, and click **Open**. The NORTHWND.MDF is installed with the product in the **Studio for Silverlight\C1.Silverlight.DataSource \Data** folder within *C:\Users\<username>\Documents\ComponentOne Samples* (Windows 7/Vista) or *C:\Documents and Settings\<username>\My Documents\ComponentOne Samples* (Windows XP).

10. Click **OK** and then click **Next**.
11. In the **Choose Your Database Objects** window, select **Tables** and click **Finish**.

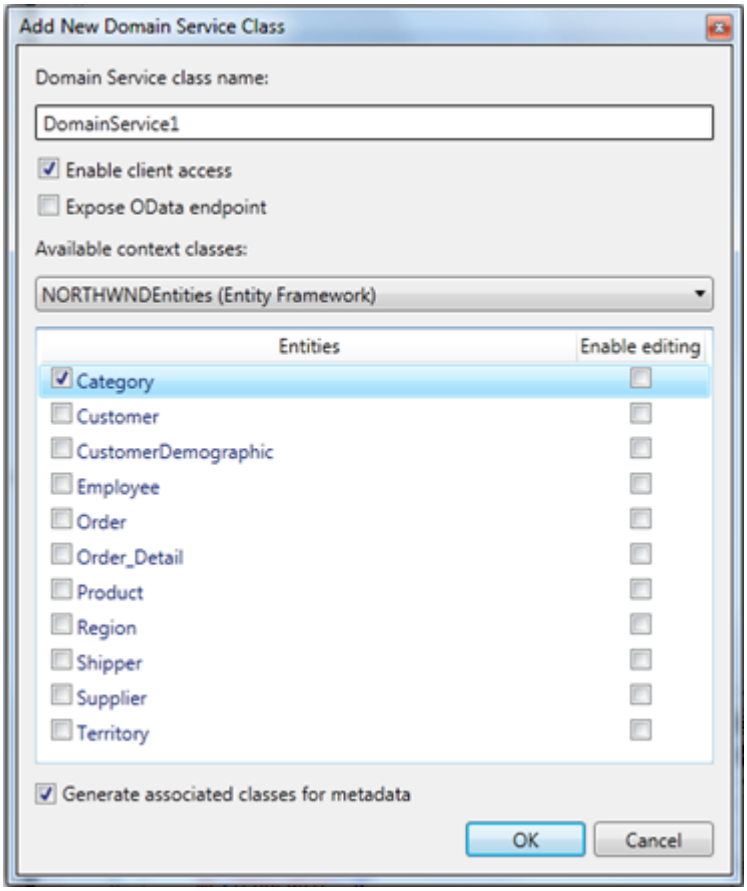


Now build the project so the new Entity Data Model classes are generated and become available throughout the project. Once they are available, you will be able to connect the data to the C1DataSource component.

Step 2 of 5: Adding a Domain Service Class

In this step, you'll add a domain service class to the project and connect it to the *Categories* table of the data source.

1. Right-click the Web project in the Solution Explorer and select **Add | New Item**.
2. Under **Installed Templates** choose the **Web** category, select **Domain Service Class** and click **Add**.
3. In the **Add New Domain Service Class** dialog box, make sure that the Entity Data Model that you just created is selected as the available DataContext class. If it is not available, go back and build the solution.
4. Check the entity types (tables) you want to use on the client, for this example choose *Category*, and click **OK**.

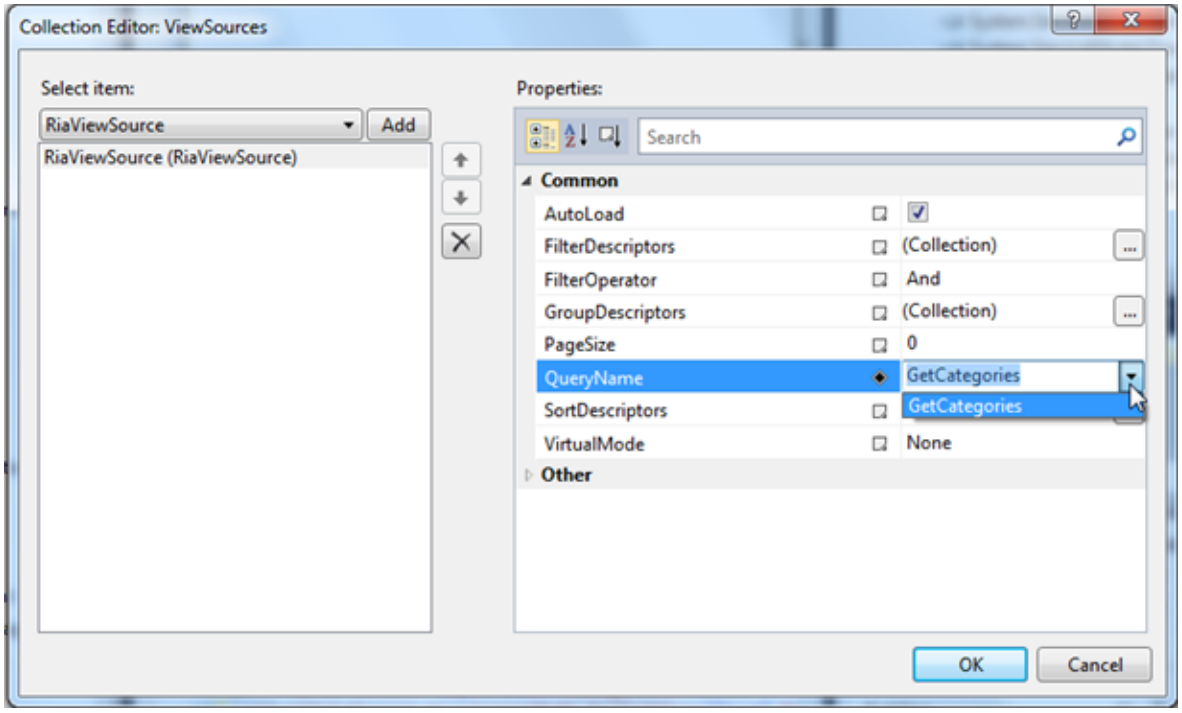


5. Build the solution. This will add the domain service class to both the Web and the client projects. To see it in the client project, select the client Silverlight project in the Solution Explorer and click **Show All Files** on the Solution Explorer toolbar. The domain service code is in the **Generated_Code** folder.

Step 3 of 5: Connecting the Data to C1DataSource

In this step, you'll add a C1DataSource component to the page and connect it to the *Categories* table of the data source.

1. Drag a C1DataSource component from the Toolbox onto the page and name it **c1DataSource1**. This is a non-visual component, so it can be placed anywhere within the page's inner content.
2. In the **Properties** window, set the C1DataSource's **DomainContextTypeName** property to the available item in the drop-down list. It should be something similar to *AppName.Web.DomainService1*.
3. Click the **ellipsis** button next to the **ViewSources** property to open the **ViewSources Collection Editor**.
4. Click **Add** and set the **QueryName** property to *GetCategories*.



5. Click **OK** to close the editor.

   With the database connected to C1DataSource, all you need to do is add a grid to display the data.

Step 4 of 5: Adding a Grid

In this step you will add a grid that will be used to display the data in the *Categories* table of the Northwind database. You can use **C1FlexGrid** or any grid you are familiar with, but in this example, we'll use a **DataGrid** control.

1. Drag a **DataGrid** control from the **Toolbox** onto your page.
2. Specify a binding for the **DataGrid**'s **ItemsSource** property in XAML:
3.
4. `ItemsSource="{Binding [GetCategories], ElementName=c1DataSource1}"`
5. Set the **DataGrid**'s **AutoGenerateColumns** property to True; otherwise, the grid will not have any columns at run time.

   Now simply run the project to view the grid!

Step 5 of 5: Running the Project

Press **F5** to run the project. The data from the *Categories* table of the Northwind database is displayed.

CategoryID	CategoryName	Description
1	Beverages	Soft drinks, coffees, teas, be
2	Condiments	Sweet and savory sauces, re
3	Confections	Desserts, candies, and sweet
4	Dairy Products	Cheeses
5	Grains/Cereals	Breads, crackers, pasta, and
6	Meat/Poultry	Prepared meats
7	Produce	Dried fruit and bean curd
8	Seafood	Seaweed and fish

ComponentOne Entity Framework DataSource Introduction

In designing the Entity Framework, Microsoft set out to create a platform agnostic means to allow developers to communicate easily with the database that underpins their desktop or server applications, and with WCF RIA Services, this principle was further extended to the Silverlight platform. Each of these frameworks provides developers with an almost ideal solution to promote data persistence (the controlled retrieval and return of data to an underlying database) but falls short of providing them with an easy way to create the application logic and interaction with bound GUI controls that forms such an intrinsic part of most applications that are being built nowadays. **ComponentOne Entity Framework DataSource** (**EF DataSource**) has been specifically designed to meet this shortfall. It enhances both frameworks by providing additional functionality and improving overall application performance. In short, it provides developers with the means to speed up the development of typical line of business applications and write less code to achieve their goal.

The following topics will examine these performance-enhancing features of the **EF DataSource** in more detail, but we'll begin by examining the critical improvements it makes to the two core frameworks.

Unified Data Context

In both the Entity Framework and WCF RIA Services, the developer is solely responsible for managing the contexts (sessions), creating them explicitly and often creating individual ones for each form in an application. Objects retrieved by one context bear no relation to those retrieved by another, even if they are essentially similar. This can create severe problems when different forms (or even different tabs on a tab control in a single form) need to interact. Up until now, the traditional way to solve these problems has been to repeatedly save data back to the underlying

database and continually refresh the GUI to reflect any changes that have occurred. The result - a lot of repeated code and degradation in overall application performance brought about through repeated trips to the database server.

ComponentOne Entity Framework DataSource (**EF DataSource**) provides a better solution by combining a unified data context and intelligent client cache. This means that an application needs only one data context which is available to all forms and controls therein. The intelligent client cache keeps track of all changes made to the context's objects, in essence enabling client-side transactional functionality, removing the need to continually save and refresh views, so application performance is improved and code simplified. Local data cache also enables client-side queries with full LINQ capabilities (filtering, sorting, grouping, and more) which would otherwise be impossible.

Virtual Data Access

With its unique virtual mode feature, **ComponentOne Entity Framework DataSource** (**EF DataSource**) supports access and data binding to large data sets, ranging from thousand to millions of rows, without awkward paging. In this mode, **EF DataSource** automatically retrieves rows from the database when they are needed for display in bound controls and then disposes of them when they are no longer required. That way the large data set appears to be in memory and ready for data binding without additional code and without consuming more memory resources than is necessary for displaying the rows currently shown by the bound controls.

The most common way of dealing with large data sets is to use paging. Paging usually provides for a poor user experience compared with data grids and other GUI controls directly bound to the data source. Virtual mode makes it possible to bind rich GUI controls directly to large data sets without paging.

More Powerful Data Binding

Out-of-the-box data binding support, both in Entity Framework and in RIA Services, is limited to binding to the same query result that was originally retrieved from the server. In other words, you can neither bind to a subset of, nor reshape in any way, the original query. Also, an out-of-the-box data source control (**DomainDataSource**) is available only for RIA Services in Silverlight, and it has serious limitations such as disallowing paging and filtering unless all data changes are committed to the database.

ComponentOne Entity Framework DataSource (**EF DataSource**) provides data source controls for direct access to Entity Framework, for both WPF and for WinForms, as well as for RIA Services (free from the standard DomainDataSource limitations).

Apart from the data source controls, **EF DataSource** supports creating live views that allow easy declarative reshaping of collections for two-way live data binding, which includes not just sorting and filtering but also calculated fields and any LINQ operations. Using live views, developers can

express a significant part, if not all, of application logic with declarative data binding producing shorter and more reliable code. Also, by defining live views over entity collections, a developer can create a view model layer in the application making it follow the popular MVVM (model-view-view model) pattern with very little code for creating a view model and no code at all for synchronizing it with the model, a task that without **EF DataSource** would require extensive manual coding.

ComponentOne Entity Framework DataSource in WinForms

ComponentOne Entity Framework DataSource (EF DataSource) includes a **C1DataSource** component which facilitates Rapid Application Development by allowing developers to specify data sources on a rich designer surface that requires little or no additional coding. In addition, the **C1DataSource** includes classes that provide support for the same features that can be controlled through the designer surface, plus many extra features that provide greater control over it through code.

We'll begin our exploration of the **C1DataSource** component by looking at how we can control it via the designer surface. From there, we'll explore how it can be controlled dynamically at run time. Not all features available at run time will be represented here. You can consult the "Programming Guide" of this documentation and the Reference for more runtime features, including client-side transaction support and more.

Simple Binding

We'll begin by creating a new Windows Forms project in Visual Studio.

1. In Visual Studio, choose **File | New Project.**
2. Select the **Windows Forms Application** template and click **OK**.

Next, add an Entity Data Model based on the Northwind database. This model will provide the data for the entire application:

1. In the **Solution Explorer**, right-click the project name and select **Add | New Item**.
2. In the **Data** category, select the "ADO.NET Data Model" item and then click **Add**.
3. Use the **Entity Data Model Wizard** to select the database you want to use. In this example, you will use "NORTHWND.mdf", which should be installed in the **Studio for WinForms\C1DataSource\Data** folder.

Now build the project so the new Entity Data Model classes are generated and become available throughout the project.

Next, add a **C1DataSource** component to the application and connect it to the Entity Data Model:

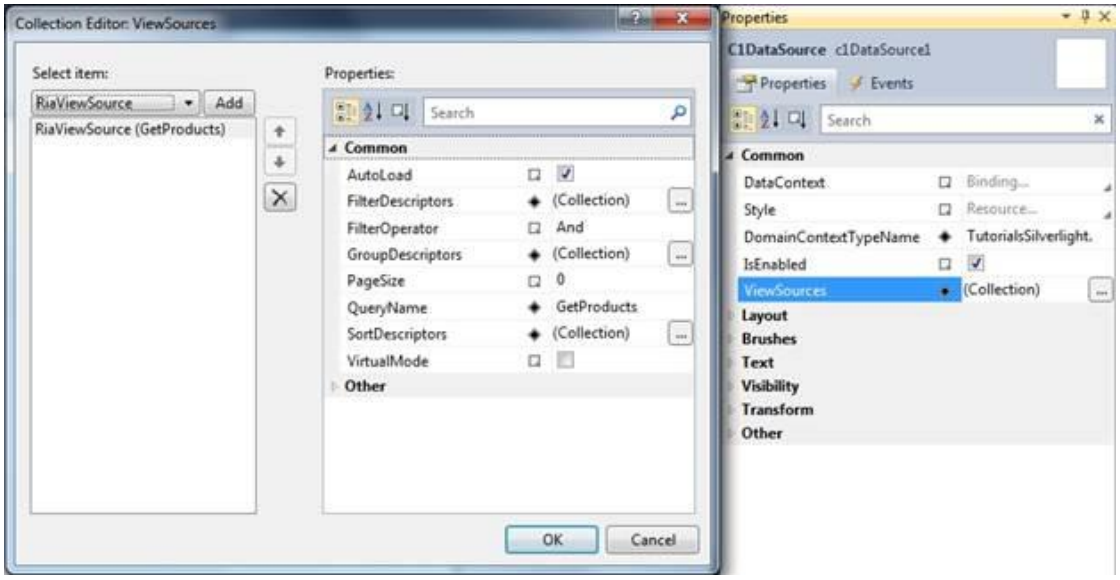
1. Drag a **C1DataSource** component from the Toolbox onto the form. This is a non-visual component, so it will appear on the tray below the form area rather than on the form itself.
2. Select the new component and choose **View | Properties Window**.
3. In the Properties window, set the **ObjectContextType** property to the type of object context you want to use. In this case, there should be only one option in the drop-down list, something similar to "AppName.NORTHWINDEntities".

At this point, the **C1DataSource** has created an application-wide object (an **EntityDataCache**) that represents the Northwind database and has application scope. Additional **C1DataSource** objects on other forms will share that same object. As long as they are part of the same application, all **C1DataSource** objects share the same **ObjectContext**.

This unified object context is one of the main advantages **EF DataSource** provides. Without it, you would have to create multiple object contexts throughout the application, and each would have to be synchronized with the others and with the underlying database separately. This would be a non-trivial task, and any errors could compromise the integrity of the data. The unified object context handles that for you transparently. It efficiently caches data and makes it available to all views in a safe, consistent way.

Now that our **C1DataSource** has an **ObjectContext** to work with, we will go on to specify the entity sets it will expose to the application through its **ViewSources** collection. Note that if you are familiar with ADO.NET, you can think of the **C1DataSource** as a **DataSet** and the **ViewSources** collection as **DataView** objects.

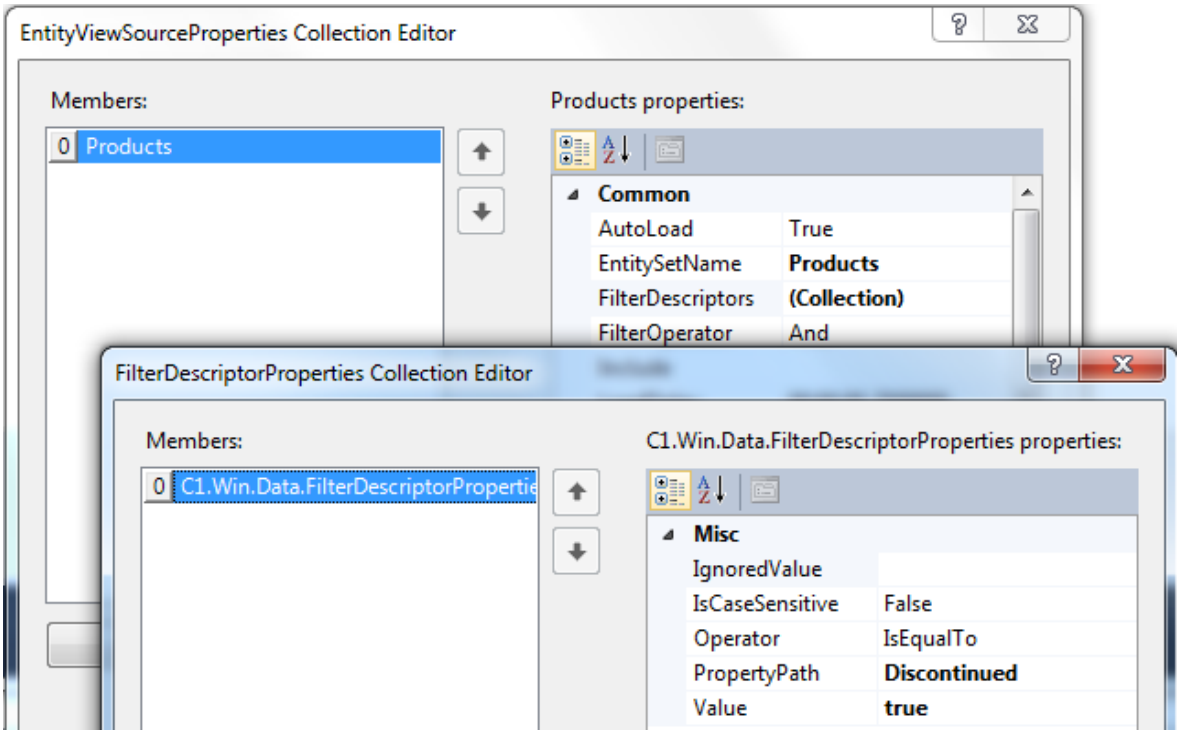
In the properties of the **C1DataSource**, locate the **ViewSourcesCollection** property and open its editor dialog. Click **Add** and then select *Products* from the **EntitySetName** drop-down list.



For this simple example, *Products* is all that is really necessary, but we could continue to create **ViewSources** within this **C1DataSource** in exactly the same way. We might, for example, create a **ViewSource** based on the *Categories* entity set allowing us to have a form that would be used to show the master-detail relationship between *Categories* and their *Products*. Conversely, there is no need to define all of the **ViewSources** that you might need in one single **C1DataSource**. You could have a separate **C1DataSource** for each **ViewSource** that you need. All you need to do is ensure that the **C1DataSource** components that you use utilize the same **ObjectContextType**.

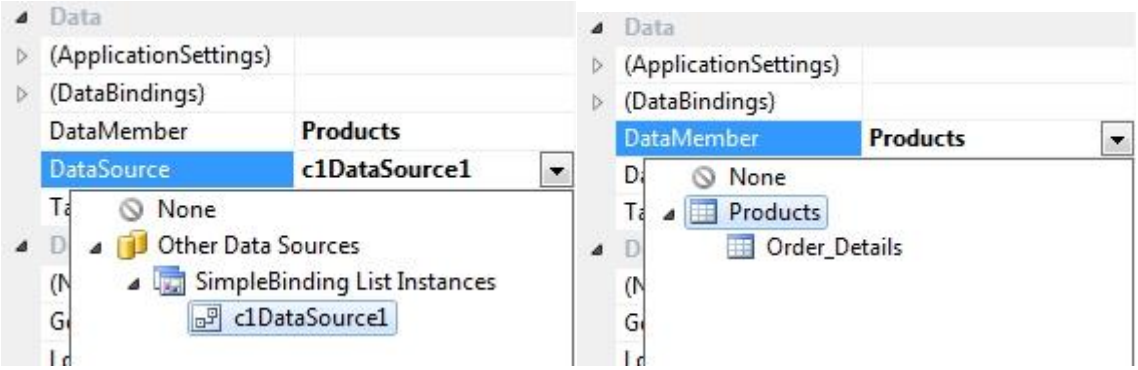
In reality we'll only want to bring a small subsection of the data contained within our database back to the client at any one time. This avoids overloading the client and network and also ensures that we are only presenting data that is relevant to the task our end user is engaged in. The traditional approach to this has been to create code (typically SQL queries) that we would run against the database to achieve our desired results. With **C1DataSource**, we can make use of the designer surface to achieve our goal without the need to write code by specifying server-side filters as property settings.

From the **ViewSourceCollection** editor, open the **FilterDescriptor** collection editor, add a filter descriptor and type the property name and a value for it for server-side filtering. If you need to filter more than one property, you can add additional filter descriptors.



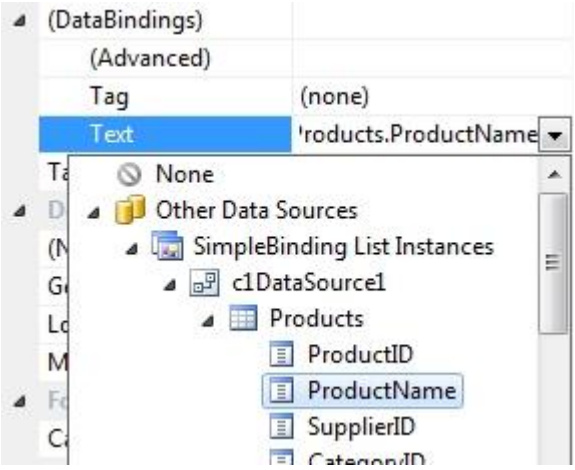
Using exactly the same methodology, you can add **SortDescriptors** provide sorting on the data you retrieve.

With our **C1DataSource** configured, add a grid control to the form. This can be **DataGridView**, a **C1FlexGrid** or indeed any grid that you are familiar with. Set the grid's **DataSource** property to the name of the **C1DataSource** (if you haven't specifically named the **C1DataSource,** then this will be *c1DataSource1*) and its **DataMember** property to *Products*. The **DataMember** property will, in fact, display a drop-down list of all the **ViewSources** (or Entity Sets) that we defined for the **C1DataSource**.



At this point, the grid will automatically generate columns for all the fields in the *Product* type, and most grids will allow you to further customize these columns and their layout via their built-in designers. Once you are happy with the grid's layout, save, build, and run the application. Notice that the data loads automatically and that you can sort, add and remove items as you'd expect to be able to do. All this has been achieved by adding just two items to your form (a **C1DatSource** and a data grid) and setting a few properties. You did not have to write a single line of code!

You could continue to add more controls to this form and bind them to specific items in the *Products* collection. To illustrate this point, add a **TextBox** control to the form. From the Properties window, expand the **DataBindings** section and bind its **Text** property to the *ProductName*, as illustrated.



Save, build and run the application again, and this time notice how the name of the product currently selected in the grid appears in the **TextBox** that you have just added to the form. If you then edit the product name in either control, the change will be immediately reflected in the other.

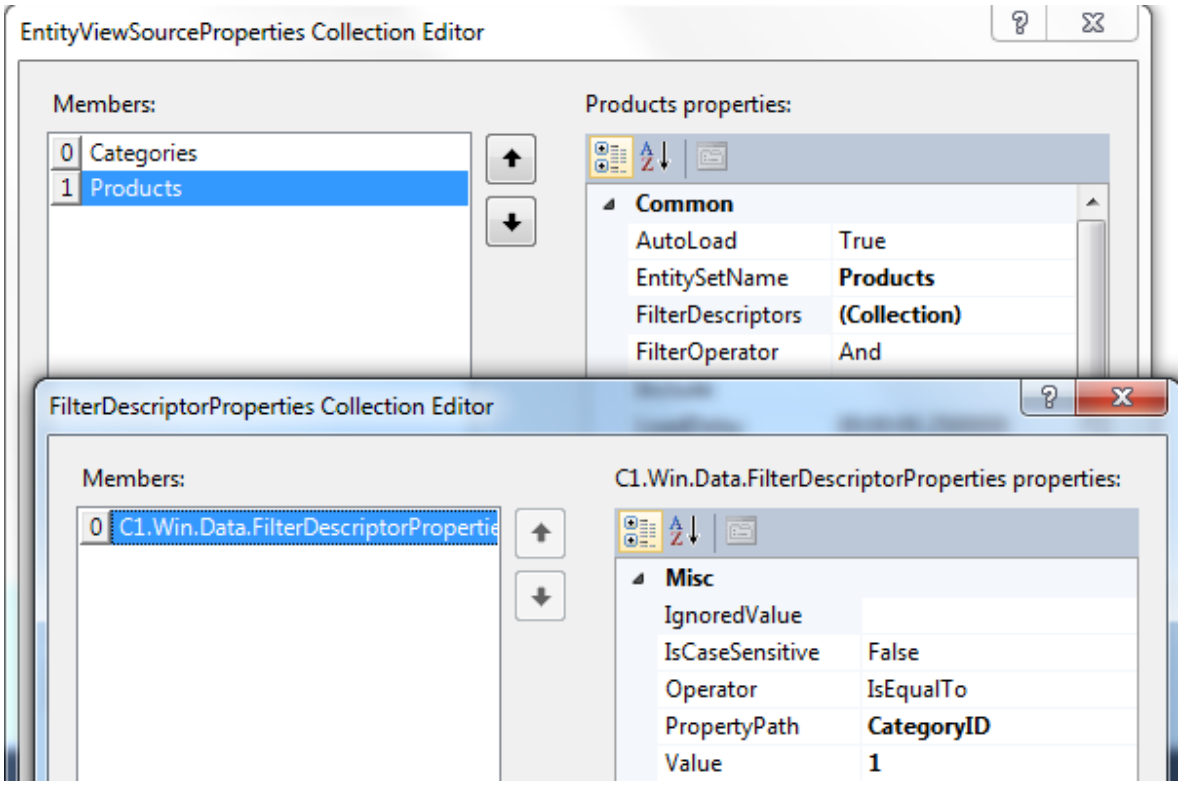
Server-Side Filtering

We already mentioned that it is generally desirable to restrict the data returned from the server to the client, and we demonstrated how C1DataSource facilitates this with the use of the **FilterDescriptor Collection Editor**. Now we'll demonstrate how to provide the end user with the means to achieve server-side filtering.

The user will select a *Product Category* from a combo box, for example, although other GUI controls can be used, and that will load a **DataGrid** with new data from the server.

To implement server-side filtering, follow these steps:

1. Add a new form with a **C1DataSource** component to the project used in Simple Binding. You can make this form the project's start up form to save time when you run the project.
2. Establish the **C1DataSource** as before, but this time define two view sources: *Categories* and *Products*. Click the **Add** button twice in the **ViewSourceCollection**. Enter *Categories* next to the EntityViewSourceProperties.EntitySetName property for the first member (0) and enter *Products* for the second member (1). For *Products*, we'll define a filter as shown in the following picture:



- Add a **ComboBox** control to the form and set the following properties:
  - DataSource = C1DataSource Name (typically C1DataSource1)
  - DisplayMember = *Categories.CategoryName*
  - ValueMember = *Categories.CategoryID*
- Add a **DataGrid** control to the form and set the following properties:
  - DataSource = C1 DataSource Name (typically C1DataSource1)
  - DataMember = *Products*
- Add the following code to the form to handle the combo box's *SelectedValueChanged* event:

To write code in Visual Basic

Visual Basic

```
Private Sub comboBox1_SelectedIndexChanged(sender As Object, e As EventArgs)
    C1DataSource1.ViewSources("Products").FilterDescriptors(0).Value =
comboBox1.SelectedValue
 End Sub
```

To write code in C#

C#

```
private voidcomboBox1_SelectedValueChanged(object sender, EventArgs e)
 {
    c1DataSource1.ViewSources["Products"].FilterDescriptors[0].Value =
       comboBox1.SelectedValue;
 }
```

- Save, build and run the application. Select a category in the combo box and notice how products related to that category are displayed in the grid. Switch between categories and notice how the data is returned. As before, all the items in the grid are editable.

The Power of Client Data Cache

The Server-Side Filtering example shows how the **ComponentOne Entity Framework DataSource** (**EF DataSource**) improves and simplifies working with the Entity Framework in applications, made possible by its client-side data cache, a key feature of **EF DataSource**. It enables several important enhancements and makes writing application code much easier.

Let's start with a performance enhancement that can be seen in the Server-Side Filtering example. When the user switches between categories, the grid loads products associated with them. The first time a category is chosen, there is a slight delay when the relevant data is retrieved. On subsequent occasions, when the same category is chosen, data retrieval is virtually instantaneous. Why? Because the data is being retrieved from the client memory data cache (EntityDataCache) rather than the server.

Note: The EntityDataCache is actually smarter still, determining that a return trip to the server can be avoided, even in more complex cases where the queries may not be identical but can be fulfilled from the results of other queries already contained therein.

This performance enhancement may not be obvious if you are working on a single machine where no network interaction is required, but in real world applications it can make all the difference between a sluggish application that calls the server on every user action and a crisp interface with no delays.

The second important enhancement is in memory management. You might think based on what you've read and observed to date that the EntityDataCache continually accumulates data throughout its existence. If this were the case you would very quickly witness severe performance degradation. In reality the EntityDataCache is keeping track of what is stored within it and as data

is no longer required is releasing it performing a self-cleansing operation at the same time. All of this is being done without the need for you to add extra code, and more importantly still it is ensuring that data integrity is being preserved at all times. Data won't be released if required by other data, nor will data that has been altered in some way without those alterations having being saved. This also relates to performance, because getting rid of unnecessary objects in memory improves performance; and vice versa, keeping large numbers of obsolete objects in memory leads to performance degradation.

We mentioned before how **EF DataSource** simplifies context management by eliminating the need to create multiple data contexts thanks to the client cache. Now we should explain what the EntityDataCache actually is and how we can further use this information to our advantage.

The EntityDataCache is essentially the context. In terms of **EF DataSource** namespaces, the cache is the **C1.Data.Entities.EntityClientCache** class, and it is in one-to-one correspondence with **ObjectContext** through its **ObjectContext** property. Both the cache and its underlying **ObjectContext** are created for you automatically if you use the **C1DataSource** component ; however, you can create them explicitly and set the **C1DataSource.ClientCache** property in code, if necessary.

To see how simple application code becomes thanks to the client-side cache, let's add the functionality of saving modified data to the Server-Side Filtering project.

1. Simply add a button, **btnSaveChanges**, to the form and add a handler for the code:

   To write code in Visual Basic

   Visual Basic

```
Private Sub btnSaveChanges_Click(sender As System.Object, e As System.EventArgs)
    C1DataSource1.ClientCache.SaveChanges()
 End Sub
```

   To write code in C#

   C#

```
private void btnSaveChanges_Click(object sender, EventArgs e)
 {
    c1DataSource1.ClientCache.SaveChanges();
 }
```

2. Save, build and run the application.
    - Select a category and then make some changes to the product information in the grid.
    - Select another category (and possibly a third) and again makes changes to the product details in the grid.
    - Click the button you added, close the application, reopen it and select the same categories as before.
    - Observe how the changes you made have been saved. **EF DataSource** has provided, via the EntityDataCache, a way to alter the product details of several categories' products without the need to save those changes each time different category is selected. To achieve this effect without **EF DataSource**, you would either need to write a lot of code, or you'd need to keep the entities (the product details) from different categories in the same context. This would waste memory without releasing it, creating a memory leak. **EF DataSource** simplifies all of this for you while optimizing memory usage and performance.

Client-side caching also makes possible other important features of **EF DataSource**, such as client-side queries and, especially, live views. Live Views is a feature that allows you to replace much of the complex application coding with simple data binding, which we'll learn more about later.

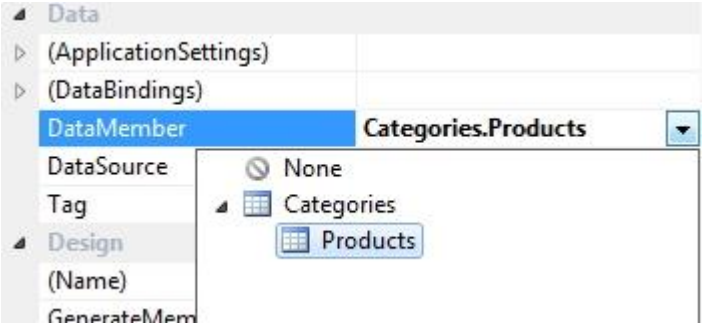
Master-Detail Binding

As we've already seen in the Server-Side Filtering example, **C1DataSource** supports master-detail binding. With very large datasets, server-side filtering is by far the best solution, but with smaller datasets, client-side filtering can be just as efficient. This next scenario demonstrates client-side master-detail binding using a grid, instead of a combo box like in our previous Server-Side Filtering example, to select our categories.

To implement master-detail binding, follow these steps:

1. Using the project we created to demonstrate Server-Side Filtering, add a new form with a **C1DataSource** component using the same **ObjectContextType** as before and create a ViewSource based on *Categories*. Note that you can make this the startup form to save time when you run the project.
2. Add what will become the 'master' grid to the form and set its **DataSource** property to the **C1DataSource** and its **DataMember** property to *Categories*.

    Now add a second grid to the form below the one you've just configured. Its **DataSource** will again be the C1DataSource, but set its **DataMember** property to the *Products* node which you'll find underneath *Categories* as shown in the following picture:



3. Save, build and run the application.

Selecting a category in the master grid causes the products linked to it to be displayed in the details grid (which as before is fully editable). Running this on a single machine, as you probably are, you won't notice any significant time lapse as you select new categories in the master grid and their respective products are displayed in the details grid. In the background, the **C1DataSource** is making use of an Entity Framework feature, implicit lazy loading, meaning that products are only being summoned for new categories as they are selected. For many scenarios, this is perfectly acceptable, but we began this section by specifically referring to master-detail relationships in small datasets. We might just as well fetch all of the products for all of the categories when the form loads and then display will be instantaneous whether on a single machine or across a network. To achieve this behavior, open the **ViewSourceCollection** editor and type *Products* in the **Include** property of the *Categories* view source.

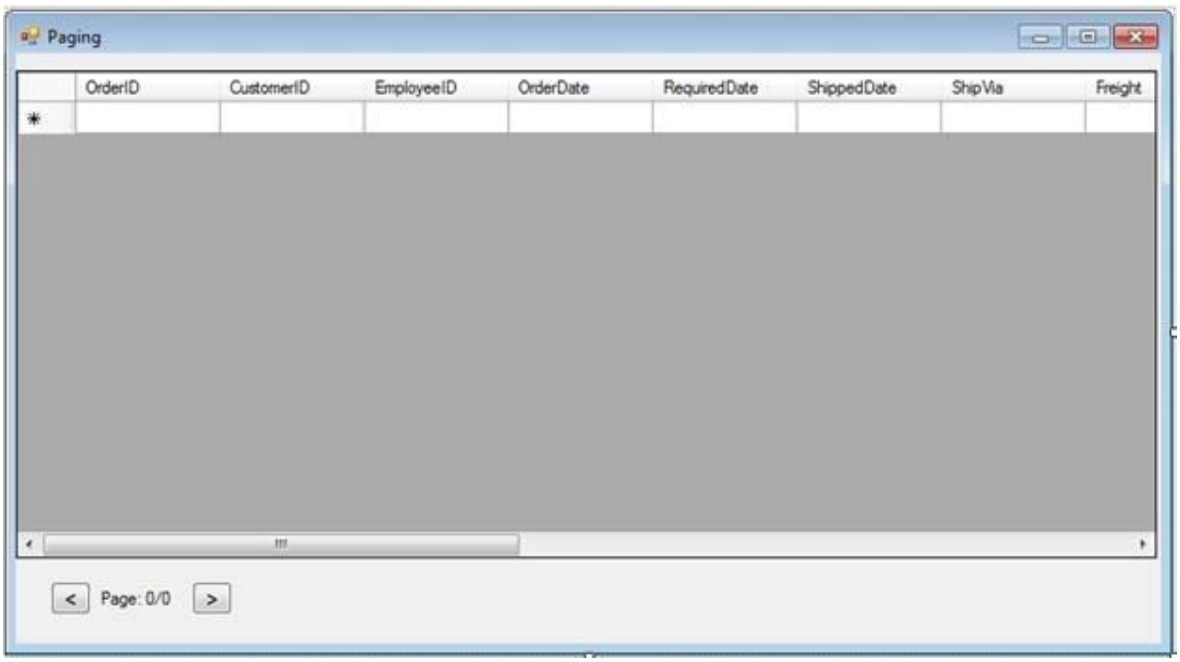
Large Datasets: Paging

To show large amounts of data without bringing it all to the client at once, applications have traditionally used paging. Paging is not an ideal solution; it complicates the interface and makes it less convenient for the user, but it is preferred in some applications. For these cases, **C1DataSource** supports paging as expected, but it does so without the traditional limitations on data modification. The user can make changes in multiple pages in one session without being forced to send the changes from one page to the database before moving to the next page. That's a substantial enhancement compared to other paging implementations, such as the one in DomainDataSource in Microsoft RIA Services.

Note: ComponentOne Entity Framework DataSource (EF DataSource) does offer a solution to the drawbacks of paging; we'll cover Virtual Mode later in this documentation.

To implement paging, follow these steps:

1. Using the project we created to demonstrate Master-Detail Binding, add a new form with a **C1DataSource** component using the same **ObjectContextType** as before. Note that you can make this the startup form to save time when you run the project.
2. Create a **ViewSource** in the **ViewSourceCollection** editor, entering *Orders* as the EntitySetName.
3. To enable paging, set the **PageSize** property to 10 for now, but you can choose any reasonable value for this property. It's simply determining the number of data rows that will be displayed on a page.
4. Add a **DataGrid** to the form and set its **DataSource** property to the **C1DataSource** and its **DataMember** property to *Orders*.
5. To facilitate easy movement between pages, add two buttons, **btnPrevPage** and **btnNextPage**, and a label, **labelPage**, as shown in the following image.



6. Add the following code containing a **RefreshPageInfo** handler for the **PropertyChanged** event used to show the current page number and page count, and handlers for button **Click** events used to move to the next and previous pages:

To write code in Visual Basic

Visual Basic

```
Imports C1.Data.DataSource

 Public Class Paging
```

```
    Private _view As ClientCollectionView

    Public Sub New()

        ' This call is required by the designer.
        InitializeComponent()

        ' Add any initialization after the InitializeComponent() call.
        _view = C1DataSource1("Orders")

        RefreshPageInfo()

        AddHandler _view.PropertyChanged, AddressOf RefreshPageInfo

    End Sub

    Private Sub RefreshPageInfo()
        labelPage.Text = String.Format("Page: {0} / {1}", _view.PageIndex + 1,
_view.PageCount)
    End Sub

    Private Sub btnPrevPage_Click(sender As System.Object, e As System.EventArgs)
Handles btnPrevPage.Click
        _view.MoveToPreviousPage()
    End Sub

    Private Sub btnNextPage_Click(sender As System.Object, e As System.EventArgs)
Handles btnNextPage.Click
        _view.MoveToNextPage()
    End Sub
 End Class
```

To write code in C#

C#

```
namespace TutorialsWinForms
 {
    public partial class Paging : Form
    {
        ClientCollectionView _view;

        public Paging()
        {
```

```
            InitializeComponent();

            _view = c1DataSource1["Orders"];

            RefreshPageInfo();
            _view.PropertyChanged += delegate { RefreshPageInfo(); };
        }

        private void RefreshPageInfo()
        {
            labelPage.Text = string.Format("Page: {0} / {1}", _view.PageIndex + 1,
_view.PageCount);
        }

        private void btnPrevPage_Click(object sender, EventArgs e)
        {
            _view.MoveToPreviousPage();
        }

        private void btnNextPage_Click(object sender, EventArgs e)
        {
            _view.MoveToNextPage();
        }
    }
}
```

7. Save, build and run the application. Page through the Orders. While you are moving between pages, try changing some data in the grid. Try changing data in one page, and then move to another page and try changing data there. Notice that **C1DataSource** allows you to do this without forcing you to save data to the database before leaving the page. This is an important enhancement compared with other paging implementations, including the one supported by DomainDataSource in Microsoft RIA Services (which is for Silverlight only, whereas **EF DataSource** supports this, as other features, for all three platforms: WinForms, WPF, Silverlight).

Try also to delete some orders. This is also allowed without restrictions, and moreover, the current page will automatically complete itself to keep the number of rows in the page unchanged. You can also add new rows. They are added to the end of the dataset.

And if you add a **Save Changes** button, using the following code as we did previously, then you will be able to save changes made in multiple pages by pressing that button when you are done.

To write code in Visual Basic

Visual Basic

```
Private Sub btnSaveChanges_Click(sender As System.Object, e As System.EventArgs)
    C1DataSource1.ClientCache.SaveChanges()
 End Sub
```

To write code in C#

C#

```
private voidbtnSaveChanges_Click(object sender, EventArgs e)
 {
   c1DataSource1.ClientCache.SaveChanges();
 }
```

All this functionality is what you would expect from a paging implementation that supports unrestricted data modification. Unfortunately, it is not easy to implement. For example, think of all possible cases where changing data on one page interferes with what should be shown on other pages, and so on. That is why paging usually imposes severe restrictions on data modifications, if they are at all allowed. For example, MS DomainDataSource requires you to save all changes before changing pages. Fortunately, **EF DataSource** supports paging without restricting data modifications in any way.

As with many other features, unlimited modifiable paging is made possible by the client-side cache, see The Power of Client Data Cache. That also means that paging implementation in **EF DataSource** is optimized both for performance and for memory consumption. The cache provides that optimization. It keeps recently visited pages in memory, so re-visiting them is usually instantaneous. And it manages memory resources, releasing old pages when necessary, to prevent memory leaks.

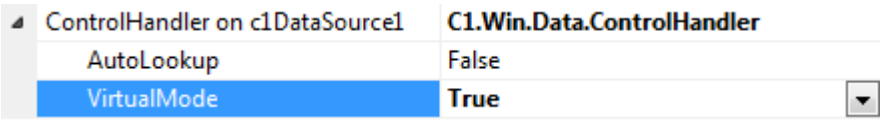
Large Datasets: Virtual Mode

As mentioned in the Large Datasets: Paging topic, **ComponentOne Entity Framework DataSource** (**EF DataSource**) has an even better solution than paging for dealing with large data and large numbers of rows.

What if using large datasets with thousands, or even millions, of rows was no longer a problem? What if you could use the same controls to display massive datasets as you would for smaller datasets, with no paging, no code changes, and all by setting one Boolean property? With **EF DataSource** you can, thanks to the magical **VirtualMode** property.

To implement virtual mode, follow these steps:

1. Using the project we created to demonstrate Large Datasets: Paging, add a new form with a **C1DataSource** component using the same **ObjectContextType** as before. Note that you can make this the startup form to save time when you run the project.
2. Create a **ViewSource** in the **ViewSourceCollection** editor. Use the largest table in our sample database: *Order_Details*.
3. Set the **VirtualMode** property to *Managed*.Another possible value is *Unmanaged*, but that is an advanced option that should be used with caution and only when necessary. The *Managed* option means that getting data from the server is managed by a grid control. With the *Managed* option, **EF DataSource** supports ComponentOne C1FlexGrid and Microsoft DataGridView. It is optimized for performance with those specific grid controls. The *Unmanaged* option means that virtual mode is not driven by a specific control, will work with any bound controls but is subject to some limitations described here.
4. Add a **DataGrid** to the form and set its **DataSource** property to the **C1DataSource** and its **DataMember** property to *Order_Details*.
5. Now, since we selected the *Managed* option, we need to specify which grid control is driving the data. Select the grid in the designer and find the property called "ControlHandler on c1DataSource1" in the Properties window, as shown in the following picture:



   This is an extender property and will be available only if there is a **C1DataSource** component on the form. Set the **VirtualMode** property there to *True*.

6. Save, build and run the application. You'll see nothing earth-shattering, simply a grid you can navigate, scroll and modify as you see fit. It looks and behaves like any conventional data grid, which is exactly the point. You can use large datasets without the drawbacks of paging and without code. And as an added benefit, you can use any GUI control you like so long as it has a **DataSource** property. This example uses a relatively modest-sized dataset, but **C1DataSource** running in virtual mode would be just as responsive with a much larger dataset; it does not depend on the number of rows. To further prove the point, look at the **OrdersDemo** sample installed with this product. The sample uses a larger database, with roughly 65,000 additional rows, but the responsiveness is the same as our example here. Again, it does not depend on the number of rows in the dataset.

How does this magic work? It works much like paging, only hiding its inner workings from the GUI controls, with paging occurring under the hood. The GUI controls see the data as if it is fetched to the client and ready for them. When GUI controls request data, **C1DataSource**, or **ClientViewSource** if it is used in code without a **C1DataSource** control, first checks whether it can serve the data from memory, from the same client-side cache it uses for all features. If it can't find it in memory, it transparently fetches the required data from the server. As with other features using the client-side cache, **C1DataSource** does not store the fetched data indefinitely, which would be a memory leak. It knows which parts of the data are needed for serving the GUI controls and which

parts should be kept because they are modified or related to modified parts, and so on. It releases old, unnecessary parts of data as needed. No code is required and any GUI controls can be used.

Automatic Lookup Columns in Grids

A common scenario in data binding is for data classes to contain references to other data classes. For example, a **Product** object may contain references to **Category** and **Supplier** objects.

In ADO.NET, the references usually appear as foreign keys that map to other tables (e.g., **Product.CategoryID** and **Product.SupplierID**).

In the Entity Framework, you still get the key columns, but you also get the actual objects. So you have **Product.CategoryID** (usually an integer) and **Product.Category** (an actual Category object).

Displaying foreign keys in a grid is not very useful, because it is unlikely that users will remember that category 12 is "Dairy Products" or that supplier 15 is "ACME Imports". Allowing users to edit these keys would be even worse. A common way to work around this problem is to remove the related entity columns from any bound grids and, optionally, to replace them with custom columns that use combo boxes for editing the values, so called lookups. The combo boxes have to be bound to the related tables and have their properties set so they display relevant values (e.g., **Category.Name** or **Supplier.CompanyName**) and so they are synchronized with the values being displayed on the grid. This is not terribly hard to do, but it is a tedious and error-prone task that makes projects harder to create and maintain.

C1DataSource can do this tedious work for the developer; it can automatically change related entity columns so that they show combo box lookups. It can do this for several types of data grids, those that it supports. Currently supported WinForms grids are: C1FlexGrid and Microsoft DataGridView. Here we will show how to do this for **C1FlexGrid**.

C1DataSource provides an *extender* property called **ControlHandler**. If you place a **C1FlexGrid** control on a form that contains a **C1DataSource**, the grid will get an additional **ControlHandler** property. A ControlHandler is an object containing (at present) a single boolean property **AutoLookup**. This property set to *True* causes the **C1DataSource** to configure grid columns that contain references to other entities so that they show lookup combos.

To see how it works, follow these steps:

1. Using the project used in Simple Binding|document=WordDocuments\C1DataStudio-WinForms.docx;topic=Simple Binding, add a new form with a **C1DataSource** component using the same **ObjectContextType** as before.
2. Create a **ViewSource** in the **ViewSourceCollection** editor, entering *Products* as the **EntitySetName**.
3. Add a **C1FlexGrid** to the form and set its **DataSource** property to the **C1DataSource** and its **DataMember** property to *Products*.
4. Save, build and run the application. It will look like this:

Product ID ▲	Product Name	Category
5	Chef Anton's Gumbo Mix	Category : 2
9	Mishi Kobe Niku	Category : 6
17	Alice Mutton	Category : 6
24	Guaraná Fantástica	Category : 1
28	Rössle Sauerkraut	Category : 7
29	Thüringer Rostbratwurst	Category : 6
42	Singaporean Hokkien Fried Me	Category : 5
53	Perth Pasties	Category : 6

As you can see, the **Category** and **Supplier** columns are not useful at all. You could remove them or customize the grid by writing some code to create new columns, but there's an easier way.

5. Select the grid in the designer and find the property called "ControlHandler on c1DataSource1" in the Properties window, as shown in the following picture:

Remember, this is an extender property and will be available only if there is a **C1DataSource** component on the form. Set the **AutoLookup** property there to *True*.

6. When you are done, run the project again and look at the **Category** and **Supplier** columns. Notice that now the grid shows the category name and supplier's company name instead of the generic strings. Also notice that you can edit the product's category and the product's supplier by picking from a drop-down list, complete with auto search functionality.

7. Finally, click the column headers to sort the grid by **Category** or **Supplier** and notice that the sorting is performed based on the value displayed on the grid. This is what anyone would expect, but surprisingly it is not easy to achieve using regular data binding.

The string value (name) shown by the combo box is determined following these rules:

1. If the entity class overrides the **ToString** method, then the string representation of the entity is obtained using the overridden **ToString** method. This should return a string that uniquely represents the entity. For example, it could be the content of a **CompanyName** column, or a combination of **FirstName**, **LastName**, and **EmployeeID**. This is the preferred method because it is simple, flexible, and easy to implement using partial classes (so your implementation will not be affected if the entity model is regenerated).
2. If the entity class does not override the **ToString** method, but one of its properties has the **DefaultProperty** attribute, then that property is used as the string representation of the entity.
3. If the entity class does not override the **ToString** method and has no properties marked with the **DefaultProperty** attribute, then the first column that contains the string "Name" or "Description" in its name will be used as a string representation for the entities.
4. If none of the above applies, then no lookup is created for the given entity type.

Customizing View

In many situations, you may want to use custom views that do not correspond directly to the tables and views provided by the database. LINQ is the perfect tool for these situations, allowing you to transform raw data using your language of choice using query statements that are flexible, concise, and efficient. **ComponentOne Entity Framework DataSource (EF DataSource)** makes LINQ even more powerful by making LINQ query statements live. That's why its LINQ implementation is called LiveLinq. We will show the power of LiveLinq in Live Views. For now, suffice it to say that you can transform your view to shape it whatever way you want using LINQ operators without losing full updatability and bindability.

Let's try, for example, one of the LINQ operators, **Select** (also called projection), to customize the fields (properties) of our view.

To customize a view, follow these steps:

1. Using the project we created to demonstrate Paging, add a new form with a **C1DataSource** component using the same **ObjectContextType** as before. Note that you can make this the startup form to save time when you run the project.
2. Create a **ViewSource** in the **ViewSourceCollection** editor. Use the *Products* table in our sample database.
3. Add a **DataGrid** to the form, but unlike previous examples, we won't actually bind its **DataSource** at design time. We'll do this at run time in code because we want to implement a custom view.
4. Add the following directive statements to the form's code:

   To write code in Visual Basic

   Visual Basic

   ```
   Imports C1.LiveLinq.LiveViews
   ```

   To write code in C#

   C#

   ```
   using C1.LiveLinq.LiveViews;
   ```

5. In the form's Load event, add the following code to create a custom live view and bind it to the grid:

   To write code in Visual Basic

   Visual Basic

   ```
   dataGridView1.DataSource = _
       (From p In C1DataSource1("Products").AsLive(Of Product)()
                       Select New With
                       {
                           p.ProductID,
                           p.ProductName,
                           p.CategoryID,
                           p.Category.CategoryName,
   ```

```
                p.SupplierID,
                .Supplier = p.Supplier.CompanyName,
                p.UnitPrice,
                p.QuantityPerUnit,
                p.UnitsInStock,
                p.UnitsOnOrder
            }).AsDynamic()
```

To write code in C#

C#

```
dataGridView1.DataSource =
    (from p in c1DataSource1["Products"].AsLive<Product>()
        select new
            {
                p.ProductID,
                p.ProductName,
                p.CategoryID,
                CategoryName = p.Category.CategoryName,
                p.SupplierID,
                Supplier = p.Supplier.CompanyName,
                p.UnitPrice,
                p.QuantityPerUnit,
                p.UnitsInStock,
                p.UnitsOnOrder
            }).AsDynamic();
```

Here **c1DataSource1**["Products"] is a ClientCollectionView object. It is the view that is created by the view source that we set up in the designer. Note that if you need to access the view source itself in code, it is also available, as **c1DataSource.ViewSources["Products"]**. The AsLive() extension method call is needed to specify the item type of the view (*Product*) so LiveLinq operators can be applied to it. The result, c1DataSource1["Products"].AsLive<Product>(), is a View<Product>.
The C1.LiveLinq.LiveViews.View is the main class of LiveLinq used for client-side live views. LINQ operators applied to live views preserve their updatability and bindability. They are the same usual LINQ operators, but the fact that they are applied to a live view gives them these additional features that are critically important for data binding applications.

Note: AsDynamic() must be applied to this view because its result selector (the select new… code) uses an anonymous class. This is a minor LiveLinq limitation, only for anonymous classes. If you forget to add AsDynamic() to such view, you will be reminded by an exception.

6. Save, build and run the application. The grid now shows the columns we defined in the 'select' clause of our LiveLinq view. Note also that all columns are modifiable. It may not seem like a big deal; however, it is an important feature that would be difficult to implement on your own for this customized view, especially when adding and deleting rows, which, as you can see here, is also supported. This is what we were referring to by stating that LiveLinq operators preserve updatability.

Bindability is achieved because the views are always 'live'; they aren't simple snapshots of static data. To prove the point, construct an exact replica of the form that we've just built. At this stage, you might find it easier to add a menu to your application to make accessing the forms you've created easier. Save, build and run the application. This time, open two instances of the form you just created and change some data in the grid on one of the forms. Notice how the corresponding data in the grid on the other form is changed automatically. Remember, you have not had to write a single line of code for the grids in these forms to synchronize the changes that you've just made.

You will see more of what LiveLinq can do in the Live Views topic.

Working with Data Sources in Code

Up to this point, we have been setting up data sources directly on the designer surface with very little code. **ComponentOne Entity Framework DataSource (EF DataSource)** has made it very easy, but sometimes you want or need to do everything in code. **EF DataSource** makes this possible as well. Everything we did previously can be done at run time in code.

An obvious way to go about this would be to use the **ClientViewSource** object that we have, in effect, been setting up in the designer as elements of the **ViewSourceCollection** of a **C1DataSource**, given that it can be created on its own without a **C1DataSource**. We could, however, take a step further back and use a lower level class ClientView<T>. This would provide full control over loading data from the server and, since it is derived from C1.LiveLinq.LiveViews.View<T>, we can apply any LiveLinq operators to it. The ability to bind this to any GUI control whose datasource can be set to a View<T> also means that we'll end up with a fully editable view of our data.

Server-side filtering is, perhaps, the most common operation, because no one usually wants entire database tables brought to the client unrestricted. Earlier we saw how **EF DataSource** made it simple to perform without code, but now we'll try it in run-time code.

To begin using **EF DataSource** in run-time code without a **C1DataSource**, add a few lines to our project's main class to create a global client-side data cache. When we used **C1DataSource**, it was created for us behind the scenes. Now we can create it explicitly using the following code:

To write code in Visual Basic

Visual Basic	Copy Code

```
Imports C1.Data.Entities

 Public Class Program
    Public Shared ClientCache As EntityClientCache
    Public Shared ObjectContext As NORTHWNDEntities

    <STAThread()> _
    Shared Sub Main()

        ObjectContext = New NORTHWNDEntities
        ClientCache = New EntityClientCache(ObjectContext)

        Application.EnableVisualStyles()
        Application.SetCompatibleTextRenderingDefault(False)
        Application.Run(New MainForm())
    End Sub
 End Class
```

To write code in C#

C#	Copy Code

```
using C1.Data.Entities;

 static class Program
 {
    public static EntityClientCache ClientCache;
    public static NORTHWNDEntities ObjectContext;

    [STAThread]
```

```
    static void Main()
    {
        ObjectContext = new NORTHWNDEntities();
        ClientCache = new EntityClientCache(ObjectContext);

        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        Application.Run(new MainForm());
}
```

This code creates a single application-wide (static) **ObjectContext** and associates it with **EntityClientCache**. As noted previously in The Power of Client Data Cache topic, the ability to have a single context (and cache) for the entire application is a great simplification made possible by **EF DataSource**.

To perform server-side filtering in run-time code, follow these steps:

1. Add a new form using the project created to demonstrate Customizing View.
2. Add a grid (**dataGridView1**), a combo box (**comboBox1**), and a button (**btnSaveChanges**) to the form.
3. Add the following code to the form class:

   To write code in Visual Basic

Visual Basic	Copy Code

```
Imports C1.Data.Entities
 Imports C1.Data

 Public Class DataSourcesInCode

    Private _scope As EntityClientScope

    Public Sub New()

        InitializeComponent()

        _scope = Program.ClientCache.CreateScope()

        Dim viewCategories As ClientView(Of Category) = _scope.GetItems(Of Category)()

        comboBox1.DisplayMember = "CategoryName"
```

```
        comboBox1.ValueMember = "CategoryID"
        comboBox1.DataSource = viewCategories

        BindGrid(viewCategories.First().CategoryID)
    End Sub

    Private Sub BindGrid(categoryID As Integer)
        dataGridView1.DataSource =
                (From p In _scope.GetItems(Of Product)().AsFiltered(Function(p As Product)
p.CategoryID.Value = categoryID)
                 Select New With
                 {
                    p.ProductID,
                    p.ProductName,
                    p.CategoryID,
                    p.Category.CategoryName,
                    p.SupplierID,
                    .Supplier = p.Supplier.CompanyName,
                    p.UnitPrice,
                    p.QuantityPerUnit,
                    p.UnitsInStock,
                    p.UnitsOnOrder
                 }).AsDynamic()
    End Sub


    Private Sub btnSaveChanges_Click(sender As System.Object, e As System.EventArgs) Handles
btnSaveChanges.Click
        Program.ClientCache.SaveChanges()
    End Sub

    Private Sub comboBox1_SelectedIndexChanged(sender As System.Object, e As
System.EventArgs) Handles comboBox1.SelectedIndexChanged
        If comboBox1.SelectedValue IsNot Nothing Then
            BindGrid(CType(comboBox1.SelectedValue, Integer))
        End If
```

```
    End Sub
End Class
```

To write code in C#

C#	Copy Code

```
namespaceTutorialsWinForms
{
    using C1.Data.Entities;
    using C1.Data;

    public partial class DataSourcesInCode : Form
    {
        private EntityClientScope _scope;

        public DataSourcesInCode()
        {
            InitializeComponent();

            _scope = Program.ClientCache.CreateScope();

            ClientView<Category> viewCategories =_scope.GetItems<Category>();

            comboBox1.DisplayMember = "CategoryName";
            comboBox1.ValueMember = "CategoryID";
            comboBox1.DataSource = viewCategories;

            BindGrid(viewCategories.First().CategoryID);
        }

        private void comboBox1_SelectedValueChanged(object sender, EventArgs e)
        {
            if (comboBox1.SelectedValue != null)
                BindGrid((int)comboBox1.SelectedValue);
        }
```

```
        private void BindGrid(int categoryID)
        {
            dataGridView1.DataSource =
                (from p in _scope.GetItems<Product>().AsFiltered(
                    p => p.CategoryID == categoryID)
                 select new
                 {
                    p.ProductID,
                    p.ProductName,
                    p.CategoryID,
                    CategoryName = p.Category.CategoryName,
                    p.SupplierID,
                    Supplier = p.Supplier.CompanyName,
                    p.UnitPrice,
                    p.QuantityPerUnit,
                    p.UnitsInStock,
                    p.UnitsOnOrder
                 }).AsDynamic();
        }

        private void btnSaveChanges_Click(object sender, EventArgs e)
        {
            Program.ClientCache.SaveChanges();
        }

    }
}
```

4. Save, build and run your application. You should see similar results to those you saw in the Server-Side Filtering example, the only difference being that this time we've implemented it all in code.

Let's take a closer look at some of the code we've just written.

The private field **_scope** is the form's gateway to the global data cache. It is a pattern we recommend you follow in all the forms where you do not employ a **C1DataSource** component

directly, as that does this for you automatically. It ensures that the entities the form needs stay in the cache while the form is alive, and that they are automatically released when all forms (scopes) holding them are released.

Creating a view showing all categories for the combo box is simple:

To write code in Visual Basic

Visual Basic	Copy Code
Dim viewCategories As ClientView(Of Category) = _scope.GetItems(Of Category)()	

To write code in C#

C#	Copy Code
ClientView<Category> viewCategories = _scope.GetItems<Category>();	

To create the view to which the grid is bound that only provides those products associated with the chosen category in the combo box required one additional operator; AsFiltered(<predicate>).

To write code in Visual Basic

Visual Basic	Copy Code
From p In _scope.GetItems(Of Product)().AsFiltered(Function(p As Product) p.CategoryID.Value = categoryID)	

To write code in C#

C#	Copy Code
from p in _scope.GetItems<Product>().AsFiltered(p => p.CategoryID == categoryID)	

Note that when this query is executed, the result does not necessarily require a round trip to the server to retrieve the products requested. The cache is examined first to see if it already contains the requested data, either because the required data has already been requested once before within this form or from another form in the application. Or, possibly a completely separate query run elsewhere in the application had requested that all products be returned, so the cache would already have all the product data. Again, this is a fundamental strength of **EF DataSource**. By providing your application with a global cache of data, its performance is continually improved throughout its lifetime.

Here we chose to create a new view, and bind the grid to it, every time the user selects a new category in the combo box (see the combo box's **SelectedValueChanged** event). However, we could have avoided the need to create new views all the time and instead created one single view using a special **BindFilterKey**, which we'll learn more about in the Simplifying MVVM topic.

So, in summary, we replicated in code what we did on the design surface with **C1DataSource** in Server-Side Filtering. We have even thrown in a little extra; we customized the fields shown in the grid columns as we did in Customizing View by adding a **Select** to our LiveLinq statement:

To write code in Visual Basic

Visual Basic	Copy Code

```
Select New With
{
    p.ProductID,
    p.ProductName,
    p.CategoryID,
    p.Category.CategoryName,
    p.SupplierID,
    Supplier = p.Supplier.CompanyName,
    p.UnitPrice,
    p.QuantityPerUnit,
    p.UnitsInStock,
    p.UnitsOnOrder
}
```

To write code in C#

C#	Copy Code

```
select new
{
    p.ProductID,
    p.ProductName,
    p.CategoryID,
    CategoryName = p.Category.CategoryName,
    p.SupplierID,

   Supplier = p.Supplier.CompanyName,

   p.UnitPrice,
    p.QuantityPerUnit,
    p.UnitsInStock,
    p.UnitsOnOrder
 };
```

Had we just wanted the raw product data returned from the table without any special formatting, we could have simply said;

```
select p;
```

Live Views

Live views is a powerful feature, so let's take a little more time to see what live views can do. Live views are designed to make data binding more powerful, so powerful, in fact, that you can develop virtually entire applications with just LINQ (in its LiveLinq form) and data binding.

Live views, called **LiveLinq**, a part of **ComponentOne Entity Framework DataSource** (**EF DataSource**), is a client-side, in-memory feature applicable not only to Entity Framework and RIA Services data, but to any observable collections in memory, including XML (LINQ to XML object model) and ADO.NET DataSets.

So, for example, you can use live views over Entity Framework data and some other data (for example, XML retrieved from some web service) to integrate that data and to provide easy full-featured data binding to the integrated data. This is a powerful tool for building applications that get data from various sources. For now we're going to concentrate on how we can use it with the Entity Framework, but if you'd like to explore **LiveLinq** in greater depth, see the ComponentOne LiveLinq documentation.

In fact, we already saw an example of such customization in Customizing View. But there we only changed the properties (fields) of the view and only applied one LINQ operator, **Select**. Let's apply some more LINQ operators to transform the view. What we do here will be similar to what we did in Customizing View, but instead of using **C1DataSource**, we'll be doing everything in code.

To use live views, follow these steps:

1. Add a new form using the project created to demonstrate Working with Data Sources in Code, and add a data grid, **dataGridView1**, to the form.
2. In the form's Load event, add the following code. Here we are following the pattern recommended in Working with Data Sources in Code.

   To write code in Visual Basic

   Visual Basic

   ```
   Private _scope As EntityClientScope = Program.ClientCache.CreateScope()
   ```

   To write code in C#

   C#

   ```
   private EntityClientScope_scope = Program.ClientCache.CreateScope();
   ```

3. Getting *Products* data from the scope, we create a live view and bind the grid to that view in the form's constructor:

   To write code in Visual Basic

   Visual Basic

   ```
   _viewProducts =
       (From p In _scope.GetItems(Of Product)()
        Where Not p.Discontinued And p.UnitPrice >= 30
        Order By p.UnitPrice
        Select New With
        {
           p.ProductID,
           p.ProductName,
           p.CategoryID,
   ```

```
        p.Category.CategoryName,
        p.SupplierID,
        .Supplier = p.Supplier.CompanyName,
        p.UnitPrice,
        p.QuantityPerUnit,
        p.UnitsInStock,
        p.UnitsOnOrder
    }).AsDynamic()
dataGridView1.DataSource = _viewProducts
```

To write code in C#

C#

```
_viewProducts =

   (fromp in _scope.GetItems<Product>()

    where!p.Discontinued && p.UnitPrice >= 30

    orderbyp.UnitPrice

    select new

    {

        ProductID = p.ProductID,

        ProductName = p.ProductName,

        CategoryID = p.CategoryID,

        CategoryName = p.Category.CategoryName,

        SupplierID = p.SupplierID,

        Supplier = p.Supplier.CompanyName,

        UnitPrice = p.UnitPrice,

        QuantityPerUnit = p.QuantityPerUnit,

        UnitsInStock = p.UnitsInStock,

        UnitsOnOrder = p.UnitsOnOrder
```

```
    }).AsDynamic();

dataGridView1.DataSource = _viewProducts;
```

In this example, we applied several **LiveLinq** operators: *Where*, *OrderBy*, and *Select*. We defined our view as containing products that aren't discontinued and have a unit price of at least 30, and we sorted our view by unit price.

We chose to store the view in a private field **_viewProducts** here:

To write code in Visual Basic

Visual Basic

```
Private _viewProducts As View(Of Object)
```

To write code in C#

C#

```
private View<dynamic> _viewProducts;
```

That is only because we will need it later. If we did not, we could use a local variable for the view.

Syntactically, the query that we wrote for **_viewProducts** is just standard LINQ. It could be done without **EF DataSource**, with standard LINQ to Objects, and the code would be the same, only instead of **_scope.GetItems<Product>()**, you would use something like **ObjectContext.Products**. In fact, we will try to do just that in a moment, once we run the project, to compare what we get from standard LINQ with what we get from **LiveLinq**.

4. Now run the project. You see that the grid shows the filtered set of products, "expensive" products that aren't discontinued, in the order that we specified with the columns that we specified. Note also that all columns are modifiable, and you can even add and delete rows in the grid. Also note that you can sort the grid at run time by clicking column headers.

To appreciate this full data binding support, compare it with what you would get if you did not use **EF DataSource**, but if you used standard LINQ to Objects instead. It's easy to compare; just replace **_scope.GetItems<Product>()** in the code with **Program.ObjectContext.Products**. Note that you will also need to remove the type **C1.LiveLinq.LiveViews.View** and use the 'var' keyword instead for it to compile, because it will no longer be a live view. The difference is obvious: with standard LINQ, the data in the grid is read-only, and the grid does not support sorting.

But live views offer even more great features. Standard LINQ to Objects produces snapshots of the data that cannot reflect changes in the source data, except some simple property changes, and even then under the strict proviso that you don't utilize a custom **Select** in your LINQ statement. Live Views, on the other hand, provide dynamic 'live' views that automatically reflect changes in their source data. As such, they simplify application development because you can, in most cases, rely on data binding to automate 'live' changes in views without the need to synchronize the changes in different parts of the application with code.

To see that the views are indeed 'live', open two forms side-by-side. Run your application and open up the Custom Columns form, which we built in Customizing View, and the Client Side Querying form, which we just built here. Make some changes to a product in the **CustomColumns** form and observe how they are reflected in the other form. If, for example, you were to increase the **UnitCost** of a product to above **30**, then it would automatically appear in the second form.

To see another example of how live views automatically synchronize themselves with changes in underlying data, follow these steps:

1. Add a live view member of the user control class:

   To write code in Visual Basic

   Visual Basic

   ```
   Private _seafoodProductsView As ClientView(Of Product)
   ```

   To write code in C#

   C#

   ```
   private ClientView<Product> _seafoodProductsView;
   ```

2. Add the following code to the form's constructor:

   To write code in Visual Basic

Visual Basic

```
_seafoodProductsView = _scope.GetItems(Of Product)().AsFiltered(Function(p)
p.CategoryID.Value = 8)
```

To write code in C#

C#

```
_seafoodProductsView = _scope.GetItems<Product>().AsFiltered(p => p.CategoryID == 8);
```

3. Add two buttons, named **btnRaise** and **btnCut**, to the form and add the following handlers to the form's code :

To write code in Visual Basic

Visual Basic

```
Private Sub raiseButton_Click(sender As System.Object, e As System.EventArgs)
    For Each p In _seafoodProductsView
      p.UnitPrice *= 1.2
    Next
  End Sub

  Private Sub cutButton_Click(sender As System.Object, e As System.EventArgs)
    For Each p In _seafoodProductsView
      p.UnitPrice /= 1.2
    Next
  End Sub
```

To write code in C#

C#

```
private voidraiseButton_Click(object sender, EventArgs e)
```

```
{
    foreach (var p in _seafoodProductsView)
        p.UnitPrice *= 1.2m;
}
private void cutButton_Click(object sender, EventArgs e)
{
    foreach (var p in _seafoodProductsView)
        p.UnitPrice /= 1.2m;
}
```

4. Save, build and run the application. As you press the buttons, notice how seafood products appear in the grid because their unit price is now either greater than or equal to **30** or how they disappear when their unit price falls below **30**. All of this happens automatically. You did not have to write any special code to refresh or synchronize the grid.
5. To see how live views can make almost any GUI-related code easier to write (and less error-prone), let's add a label that shows the current row count. Without **EF DataSource**, this would usually be done in a method counting the rows, and that method would have to be called in every place in the code where that count can change. In this example, there would be three such places: on initial load, and in the two methods called when the buttons are pressed, raiseButton_Click and cutButton_Click. So it is not very easy to synchronize display with changing data even in this simple example, not to speak of a real application. Live views make all this synchronization code unnecessary. We already saw how it works for a view containing filtering, ordering, and projection (Where, OrderBy, and Select). It works as well for views containing aggregation operations, such as Count. A little difference here is that regular LINQ Count returns a single number, to which you can't bind a control, so **EF DataSource** provides a special operation LiveCount that returns a live view instead of a single number, so you can use it in data binding. We can create this binding with a single line of code:
6.
7.
```
labelCount.DataBindings.Add(new Binding("Text",
                    _viewProducts.LiveCount(), "Value"));
```

Simplifying MVVM

The Model-View-View-Model (**MVVM**) pattern is gaining in popularity as developers realize the benefits it gives their applications, making them easier to maintain and test and, particularly in the case of WPF and Silverlight applications, allowing a much clearer division of labor between the designer of the UI and the creator of the code that makes it work. However, without effective tools to aid program development based on **MVVM** patterns, programmers can actually find themselves

working harder because of the need to implement an extra layer (the View Model) and ensure that data is properly synchronized between that and the Model. This extra burden isn't onerous when you have a relatively small application with just a few simple collections (and best practice with MVVM is to use **ObservableCollection** as the datasource), but the bigger it becomes and the more collections it spawns, the worse it becomes. **ComponentOne Entity Framework DataSource** (**EF DataSource**) can ease the burden.

EF DataSource lets you use live views as your view model. Just create live views over your model collections and use them as your view model. Live views are synchronized automatically with their sources (model), so you don't need *any* synchronization code - it's all automatic. And live views are much easier to create than to write your own view model classes using **ObservableCollection** and filling those collections manually in code. You have all the power of LINQ at your disposal to reshape model data into live views. So, not only does synchronization code disappear, but the code creating view models is dramatically simplified.

To demonstrate how easy it is to follow the **MVVM** pattern using live views, let's create a form combining all features from two previous examples: the **Category-Products** master-detail from Working with Data Sources in Code and the reshaping/filtering/ordering of data from Live Views. It will be a **Category-Products** view, showing non-discontinued products whose unit price is at least **30**, ordered by unit price, and displaying a customized set of product fields in a master-detail form where the user can select a category to show the products of that category. We'll follow the **MVVM** pattern. The form (view), called **CategoryProductsView**, only hosts GUI controls (a combo box and a grid) and does not have any code except what is used to set the data source to the view model, like this:

To write code in Visual Basic

Visual Basic	Copy Code

```
PublicClassCategoryProductsView

    PrivateviewModel AsCategoryProductsViewModel= NewCategoryProductsViewModel()

    PublicSubNew()
        InitializeComponent()

        comboBox1.DisplayMember = "CategoryName"
        comboBox1.ValueMember = "CategoryID"
        comboBox1.DataSource = viewModel.Categories
        dataGridView1.DataSource = viewModel.Products

    EndSub
```

```
EndClass
```

To write code in C#

C#	Copy Code

```
publicpartialclassCategoryProductsView: Form
{
    CategoryProductsViewModelviewModel = newCategoryProductsViewModel();

    publicCategoryProductsView()
    {
        InitializeComponent();

        comboBox1.DisplayMember = "CategoryName";
        comboBox1.ValueMember = "CategoryID";
        comboBox1.DataSource = viewModel.Categories;
        dataGridView1.DataSource = viewModel.Products;
    }
}
```

All logic is in the view model class, separate from the GUI. More exactly, there are three classes: **CategoryViewModel**, **ProductViewModel**, and **CategoryProductsViewModel**. The first two are simple classes defining properties with no additional code:

To write code in Visual Basic

Visual Basic	Copy Code

```
PublicClassCategoryViewModel
    PublicOverridablePropertyCategoryID AsInteger
    PublicOverridablePropertyCategoryName AsString
EndClass
```

```
Public ClassProductViewModel
    PublicOverridablePropertyProductID AsInteger
    PublicOverridablePropertyProductName AsString
    PublicOverridablePropertyCategoryID AsInteger?
    PublicOverridablePropertyCategoryName AsString
    PublicOverridablePropertySupplierID AsInteger?
    PublicOverridablePropertySupplierName AsString
    PublicOverridablePropertyUnitPrice AsDecimal?
    PublicOverridablePropertyQuantityPerUnit AsString
    PublicOverridablePropertyUnitsInStock AsShort?
    PublicOverridablePropertyUnitsOnOrder AsShort?
EndClass
```

To write code in C#

C#	Copy Code

```
public classCategoryViewModel

{

    public virtual int CategoryID { get; set; }

    public virtual string CategoryName { get; set; }

}



public classProductViewModel

{

    public virtual int ProductID { get; set; }
```

```
    public virtual string ProductName { get; set; }

    public virtual int? CategoryID { get; set; }

    public virtual string CategoryName { get; set; }

    public virtual int? SupplierID { get; set; }

    public virtual string SupplierName { get; set; }

    public virtual decimal? UnitPrice { get; set; }

    public virtual string QuantityPerUnit { get; set; }

    public virtual short? UnitsInStock { get; set; }

    public virtual short? UnitsOnOrder { get; set; }

}
```

And here is the code for the **CategoryProductsViewModel** class:

To write code in Visual Basic

Visual Basic	Copy Code

```
PublicClassCategoryProductsViewModel

    Private_scope AsEntityClientScope

    Private_categories AsBindingSource
    PublicPropertyCategories AsBindingSource
        Get
            Return_categories
        EndGet
        PrivateSet(value AsBindingSource)
            _categories = value
        EndSet
```

```
    EndProperty

    Private_products AsBindingSource
    PublicPropertyProducts AsBindingSource
        Get
            Return_products
        EndGet
        PrivateSet(value AsBindingSource)
            _products = value
        EndSet
    EndProperty

    PublicSubNew()
        _scope = Program.ClientCache.CreateScope()

        DimCategoriesView AsObject=
            Fromc In_scope.GetItems(OfCategory)()
            SelectNewCategoryViewModelWith
            {
                .CategoryID = c.CategoryID,
                .CategoryName = c.CategoryName
            }
        Categories = NewBindingSource(CategoriesView, Nothing)

        DimProductsView AsObject=
            Fromp In_scope.GetItems(OfProduct)().AsFilteredBound(Function(p)
p.CategoryID.Value).BindFilterKey(Categories, "Current.CategoryID").Include("Supplier")
            SelectNewProductViewModelWith
            {
                .ProductID = p.ProductID,
                .ProductName = p.ProductName,
                .CategoryID = p.CategoryID,
                .CategoryName = p.Category.CategoryName,
                .SupplierID = p.SupplierID,
                .SupplierName = p.Supplier.CompanyName,
                .UnitPrice = p.UnitPrice,
```

```
            .QuantityPerUnit = p.QuantityPerUnit,
            .UnitsInStock = p.UnitsInStock,
            .UnitsOnOrder = p.UnitsOnOrder
        }
        Products = NewBindingSource(ProductsView, Nothing)

    EndSub
EndClass
```

To write code in C#

C#	Copy Code

```
public classCategoryProductsViewModel

{

    private C1.Data.Entities.EntityClientScope _scope;



    publicBindingSource Categories { get; private set; }

    public BindingSource Products { get; private set; }



    public CategoryProductsViewModel()

    {

        _scope = Program.ClientCache.CreateScope();



        Categories = new BindingSource(

            from c in _scope.GetItems<Category>()
```

```
        select new CategoryViewModel()

        {

            CategoryID = c.CategoryID,

            CategoryName = c.CategoryName

        }, null);



    Products = new BindingSource(

        from p in _scope.GetItems<Product>().AsFilteredBound(p =>
        p.CategoryID).BindFilterKey(Categories,
            "Current.CategoryID").Include("Supplier")

        select new ProductViewModel()

        {

            ProductID = p.ProductID,

            ProductName = p.ProductName,

            CategoryID = p.CategoryID,

            CategoryName = p.Category.CategoryName,

            SupplierID = p.SupplierID,

            SupplierName = p.Supplier.CompanyName,

            UnitPrice = p.UnitPrice,

            QuantityPerUnit = p.QuantityPerUnit,

            UnitsInStock = p.UnitsInStock,

            UnitsOnOrder = p.UnitsOnOrder

        }, null);
```

```
    }

}
```

Basically, it contains just two LiveLinq statements, nothing more. The statements (creating live views, see Live Views) are wrapped in **BindingSource** constructors to add currency, current item, support to the **Categories** and **Products** collections exposed by the view model class. Note that using **BindingSource** is only necessary in WinForms, because the WinForms platform is not as well suited for MVVM as WPF or Silverlight. Note also that because we use **BindingSource**, you need to add the following statement to the code file (in WinForms only):

```
using System.Windows.Forms;
```

Similar to what we saw in Working with Data Sources in Code, using **AsFilteredBound**() gives us server-side filtering. We also connected the filter key, which is the **Product.CategoryID** property, to the **CategoryID** selected by the user using a combo box event. Here we can't do this because we must keep our code independent of the GUI. So we use a BindFilterKey method to bind the filter key to the **Category.CategoryID** property of the item currently selected in the **Categories** collection. This is one reason why we need to support currency in the **Categories** collection and why we wrapped it in a **BindingSource** to get currency support in WinForms.

The **Include** ("Supplier") operator is not strictly necessary; it is used here for performance optimization. Without it, **Entity Framework** will fetch **Supplier** objects lazily, one-by-one, every time the user accesses an element of the **Products** collection whose **Supplier** was not yet fetched. This can cause delays, and it's generally much less efficient to fetch data in single rows rather than in batches, so we opted out of lazy loading here using **Include**("Supplier"), which tells the **Entity Framework** to fetch supplier information in the same query with products.

Using Entity Framework DataSource in MVVM with other MVVM framew

ComponentOne Entity Framework DataSource (**EF DataSource**) can be used to build Model-View-View-Model (**MVVM**) applications with any other MVVM frameworks.

ComponentOne Entity Framework DataSource offers several features to make your MVVM development easier:

- **Simplifies MVVM programming**

  Given that **EF DataSource** can be used to simplify MVVM programming, as seen in the Simplifying MVVM topic, it's clearly a tool that can be used for MVVM.

- **Helps create view model classes and alleviate code bloating**

  There are a plethora of tools and frameworks that developers can use to aid them in their work with MVVM, but very few that can help them create view model classes. The majority are designed to help with tasks such as passing commands and messages between the view and view model. Creating view model classes and then synchronizing them with model data is left almost entirely to manual coding. This is the primary cause of code bloating in most MVVM applications and precisely the one that **EF DataSource** is designed to alleviate in a way that is entirely compatible with other frameworks.

- **Allows you to use any framework and live views**

  You can use any framework you like to assist your MVVM application development and simply call on **EF DataSource** to provide live views to help you create view model classes.

To demonstrate these important points, we provide a sample based on the code from the well-known article by Josh Smith, one of the authors of MVVM, "WPF Apps With The Model-View-ViewModel Design Pattern" (http://msdn.microsoft.com/en-us/magazine/dd419663.aspx).

The sample can be found in the **ComponentOne Samples\Entity Framework DataSource\OrdersDemo\Orders-EF-MVVM** folder using the following paths, depending on your operating system:

Windows XP path: C:\Documents and Settings\<username>\My Documents\ComponentOne Samples

Windows 7/Vista path: C:\Users\<*username*>\Documents\.

Essentially all the files in our modified sample are the same as the originals (bar a few cosmetic changes) except one (ViewModels\OrdersViewModel.cs).

In this file, we build the view model class the **EF DataSource** way, using live views. You can see how many re-shaping functions are applied to model data to construct a view model, all done exclusively through LINQ. This made it easy and required little code. The best part is that it synchronizes automatically with model data when data in either of the two layers is changed - no synchronization code was necessary.

The fact that we only changed the way that the view model classes themselves were created (they are still derived from the original base class 'ViewModelBase') and made no other changes to the framework code that Josh Smith had employed in his original sample should serve as an example that **EF DataSource** is entirely compatible with other frameworks. You can continue to use your preferred frameworks when working with MVVM, but now you have an additional tool to make your MVVM development even easier.

ComponentOne Entity Framework DataSource in Silverlight

ComponentOne Entity Framework DataSource (**EF DataSource**) includes a **C1DataSource** control that allows you to specify data sources for RIA services client with rich features in a Rapid Application Development fashion, using the familiar object model of the standard **DomainDataSource** control, but providing richer functionality and easier application development.

In addition, the **C1DataSource** includes classes that provide support for the same features that can be controlled through the designer surface, plus many extra features that provide greater control over it through code.

We'll begin our exploration of the **C1DataSource** component by looking at how we can control it via the designer surface. From there, we'll explore how it can be controlled dynamically at run time. Not all features available at run time will be represented here. You can consult the "Programming Guide" of this documentation and the Reference for more runtime features, including client-side transaction support and more.

Simple Binding

We'll begin by creating a new windows form project in Visual Studio.

1. In Visual Studio, choose **File | New | Project.**
2. Select the **Silverlight Application** template and click **OK**.
3. In the **New Silverlight Application** dialog box, select the **Enable WCF RIA Services** checkbox and click **OK**. The wizard will create two projects: one is a Silverlight project (client) and the other is a Web project (server).

Next, add an Entity Data Model based on the Northwind database. This model will provide the data for the entire application:

1. In the **Solution Explorer**, right-click the Web project name and select **Add | New Item**.
2. In the **Data** category, select the **ADO.NET Data Model** item and then click **Add**.
3. Select **Generate from database** and click **Next**.
4. Create a connection string pointing to the Northwind SQL Server database and click **Next**. In this example, you will use "NORTHWND.mdf", which should be installed in the **Studio for Silverlight\C1.Silverlight.DataSource\Data** folder.
5. Select the **Tables** node and click **Finish**.
6. Build the project so the new Entity Data Model classes are generated and become available throughout the project.

Now we'll add a Domain Service Class to the Web project.

1. Right-click the Web project in the Solution Explorer and select **Add** | **New Item**.

2. In the **Web** category, select **Domain Service Class** and click **Add**.
3. In the **Add New Domain Service Class** dialog box, make sure that the Entity Data Model that you just created is selected as the available DataContext class. If it is not available, go back and build the solution.
4. Check the entity types (tables) you want to use on the client and click **OK**.
5. Build the solution. This will add the domain service class to both the Web and the client projects. To see it in the client project, select the client Silverlight project in the Solution Explorer and click **Show All Files** on the Solution Explorer toolbar. The domain service code is in the **Generated_Code** folder.

Next, add a **C1DataSource** component to the application and connect it to the Entity Data Model:

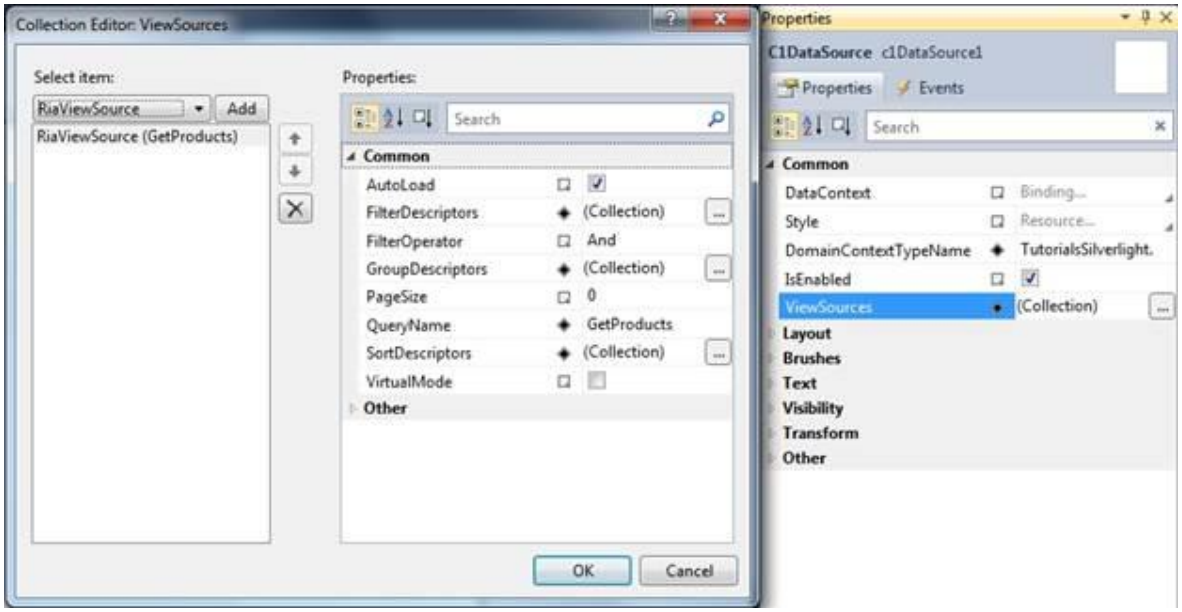
1. Drag a **C1DataSource** component from the Toolbox onto the form. This is a non-visual component, so it can be placed anywhere within the form's inner content.
2. Select the new component and choose **View | Properties Window**.
3. In the Properties window, set the **DomainContextTypeName** property to the type of **DomainContext** context you want to use. In this case, there should be only one option in the drop-down list, something similar to *MyApplication.Web.DomainService1*.

At this point, the **C1DataSource** has created an application-wide object (an **EntityDataCache**) that represents the Northwind database and has application scope. Additional **C1DataSource** objects on other forms will share that same object. As long as they are part of the same application, all **C1DataSource** objects share the same **DomainContext**.

This unified object context is one of the main advantages **EF DataSource** provides. Without it, you would have to create multiple object contexts throughout the application, and each would have to be synchronized with the others and with the underlying database separately. This would be a non-trivial task, and any errors could compromise the integrity of the data. The unified object context handles that for you transparently. It efficiently caches data and makes it available to all views in a safe, consistent way.

Now that our **C1DataSource** has a **DomainContext** to work with, we will go on to specify the entity sets it will expose to the application through its **ViewSources** collection. Note that if you are familiar with ADO.NET, you can think of the **C1DataSource** as a **DataSet** and the **ViewSources** collection as **DataView** objects.

In the properties of the **C1DataSource**, locate the **ViewSourcesCollection** property and open the collection editor. Click **Add** and then select *GetProducts* from the **QueryName** drop-down list.

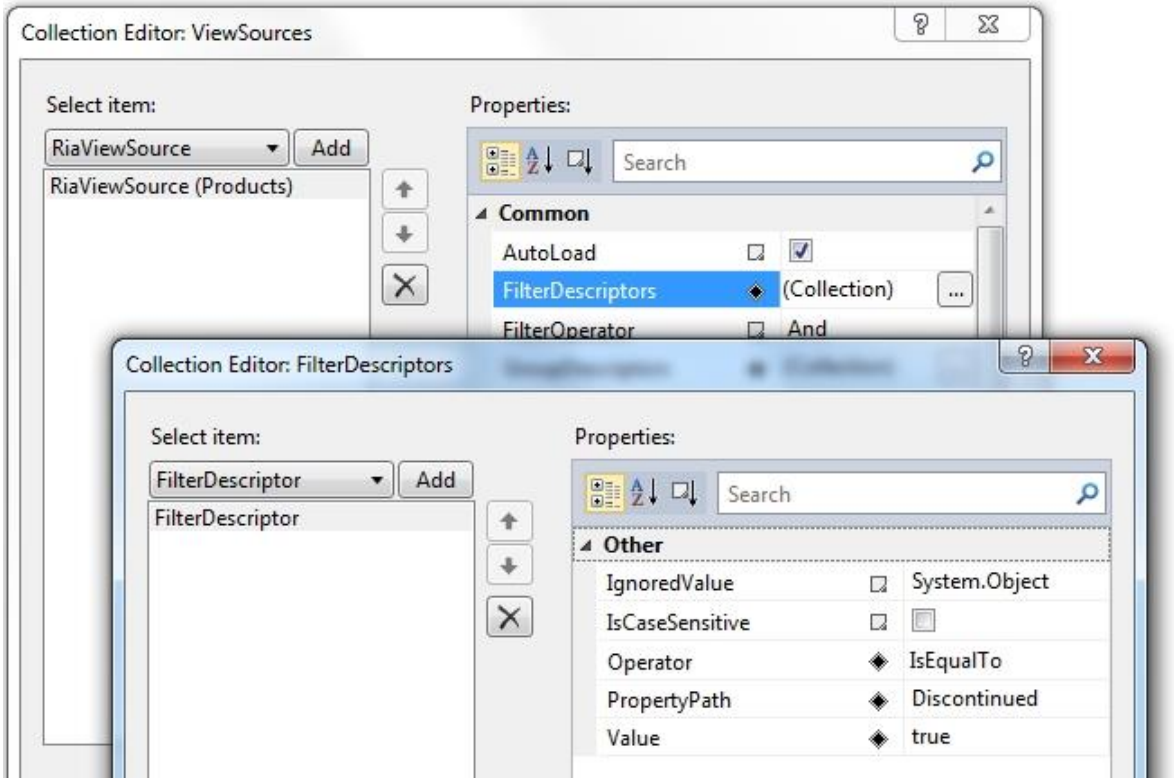


Names of the entity sets correspond to the names of the query methods defined in your domain service class. If you don't like the name, you can rename it by setting the **NameOverride** property of the view source. For example, if you don't like the "Get" prefix, you can set **NameOverride**=*Products*. It will still use the *GetProducts* query method to get data, only the view source name will change.

For this simple example, the *GetProducts* entity is all that's necessary. In other situations, like in master-detail scenarios, you can add more than one entity set. Note that you don't have to define all entity sets in one **C1DataSource**. You can have a separate **C1DataSource** for each **ViewSource** that you need. However, you must ensure that the **C1DataSource** components that you use utilize the same **DomainContextType**.

In reality we'll only want to bring a small subsection of the data contained within our database back to the client at any one time. This avoids overloading the client and network and also ensures that we are only presenting data that is relevant to the task our end user is engaged in. The traditional approach to this has been to create code (typically SQL queries) that we would run against the database to achieve our desired results. With **C1DataSource**, we can make use of the designer surface to achieve our goal without the need to write code by specifying server-side filters as property settings.

From the **ViewSourceCollection** editor, open the **FilterDescriptor** collection editor, add a filter descriptor and type the property name and a value for it for server-side filtering. If you need to filter more than one property, you can add additional filter descriptors.



Using exactly the same methodology, you can add **SortDescriptors** and **GroupDescriptors** to specify sorting and grouping of the data.

With our **C1DataSource** configured, add a grid control to the designer window. This can be a Microsoft **DataGrid**, a ComponentOne **FlexGrid**, or any other grid control you like to work with. Once the grid control is on the designer surface, specify a binding for its **ItemsSource** property in XAML:

```
ItemsSource="{Binding [GetProducts],ElementName=c1DataSource1}"
```

If you renamed the entity set to *Products* (by setting its **NameOverride** property), this will be

```
ItemsSource="{Binding [Products],ElementName=c1DataSource1}"
```

At this point, you need to either specify columns of the grid in the designer or set the grid's **AutoGenerateColumns** property to *True*; otherwise, the grid will not have any columns at run time. In real applications, you will probably specify columns in the designer, but for this example, we'll set **AutoGenerateColumns** = *True*.

Now run the project. Notice that the data loads automatically, and you can sort and modify data as you would expect. All this has been achieved by adding just two items to your form (a **C1DatSource** and a data grid) and setting a few properties. You did not have to write a single line of code!

You could continue to add more controls to this form and bind them to specific items in the *Products* or other collection. To illustrate this point, add a **TextBox** control to the form and bind its **Text** property to the *ProductName* in XAML:

in XAML:

```
Text="{Binding [GetProducts].ProductName,ElementName=c1DataSource1}"
```

Save, build and run the application again, and this time notice how the name of the product currently selected in the grid appears in the **TextBox** that you have just added to the form. If you then edit the product name in either control, the change will be immediately reflected in the other.

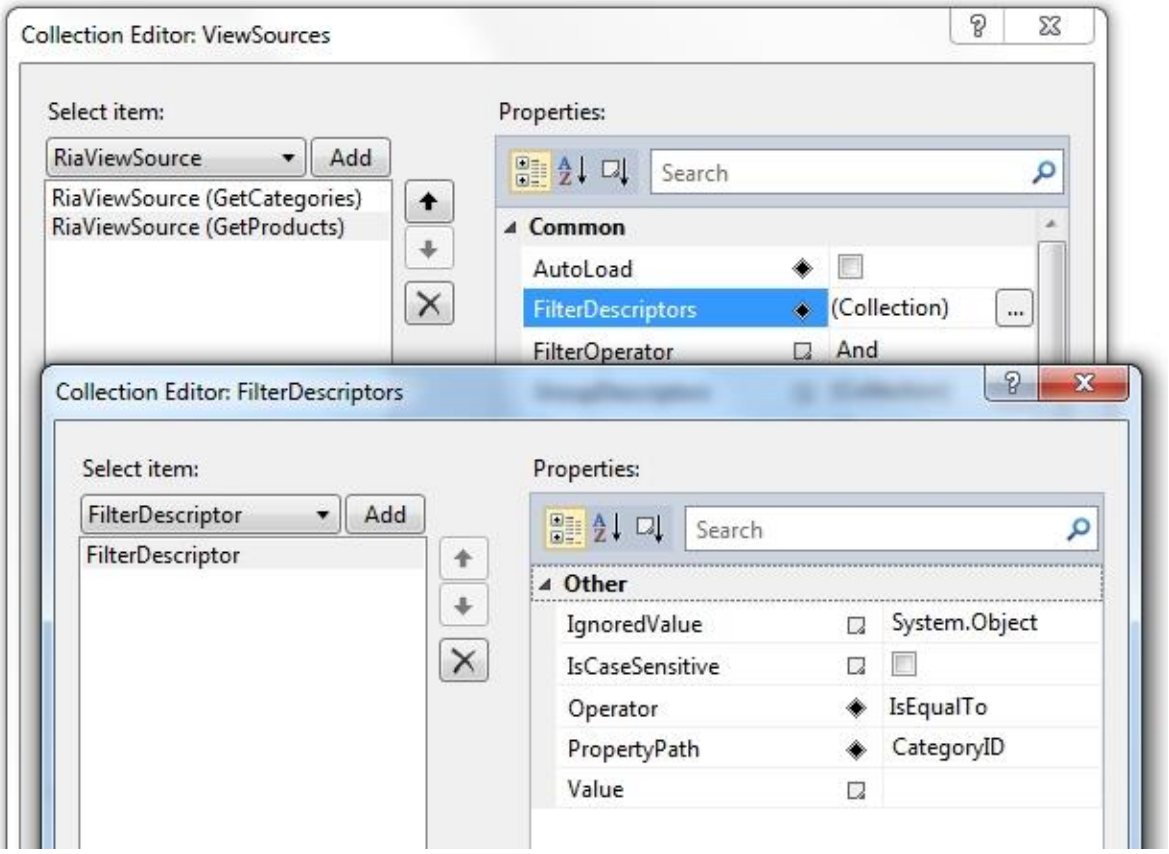
Server-Side Filtering

We already mentioned that it is generally desirable to restrict the data returned from the server to the client and we demonstrated how **C1DataSource** facilitates this with the use of the **FilterDescriptor Collection Editor**. Now we'll demonstrate how to provide the end user with the means to achieve server-side filtering.

The user will select a *Product Category* from a combo box, for example, although other GUI controls can be used, and that will load a **DataGrid** with new data from the server.

To implement server-side filtering, follow these steps:

1. Add a new form with a **C1DataSource** component to the project used in Simple Binding. You can make this form the project's start up form to save time when you run the project.
2. Establish the **C1DataSource** as before, but this time define two view sources: *GetCategories* and *GetProducts*. Click the **Add** button twice in the **ViewSourceCollection**. Enter *GetCategories* next to the **QueryName** property for the first **RiaViewSource** and enter *GetProducts* for the second **RiaViewSource**. For *GetProducts*, we'll define a filter as shown in the following picture:



Note that we left **Value** empty. This is because we will be setting it in code in response to a selection change event of the combo box.

3. Bind the grid using *ItemsSource="{Binding [GetProducts], ElementName=c1DataSource1 }"* as was shown earlier. Don't forget to set the **AutoGenerateColumns** property of the grid to *True* in XAML:

```
<sdk:DataGrid AutoGenerateColumns="True"

   ItemsSource="{Binding[GetProducts], ElementName=c1DataSource1}" />
```

And for the combo box use the following bindings:

```
ItemsSource="{Binding [GetCategories],ElementName=c1DataSource1}"

   DisplayMemberPath="CategoryName"
```

4. Finally, add the following code to the form to handle the combo box's *SelectionChanged* event:

To write code in Visual Basic

Visual Basic

```
Private Sub comboBox1_SelectionChanged(sender As System.Object, e As
System.Windows.Controls.SelectionChangedEventArgs) Handles
comboBox1.SelectionChanged
    c1DataSource1.ViewSources("GetProducts").FilterDescriptors(0).Value =
CType(comboBox1.SelectedValue, Category).CategoryID
    c1DataSource1.ViewSources("GetProducts").Load()

    End Sub
```

To write code in C#

C#

```
private voidcomboBox1_SelectionChanged(object sender, SelectionChangedEventArgs e)

 {

   c1DataSource1.ViewSources["GetProducts"].FilterDescriptors[0].Value =

       ((Category)comboBox1.SelectedItem).CategoryID;

   c1DataSource1.ViewSources["GetProducts"].Load();

 }
```

Save, build and run the application. Select a category in the combo box and notice the products that belong to that category in the grid. You can still edit the data in the grid exactly as before.

The Power of Client Data Cache

The Server-Side Filtering example shows how the **ComponentOne Entity Framework DataSource** (**EF DataSource**) improves and simplifies working with the Entity Framework and RIA Services in

applications, made possible by its client-side data cache, a key feature of **EF DataSource**. It enables several important enhancements and makes writing application code much easier.

Let's start with a performance enhancement that can be seen in the simple Server-Side Filtering example. When the user switches between categories, the grid loads products associated with them. The first time a category is chosen, there is a slight delay when the relevant data is retrieved. On subsequent occasions, when the same category is chosen, data retrieval is virtually instantaneous. Why? Because the data is being retrieved from the client memory data cache (EntityDataCache) rather than the server.

Note: The EntityDataCache is actually smarter still, determining that a return trip to the server can be avoided, even in more complex cases where the queries may not be identical but can be fulfilled from the results of other queries already contained therein.

This performance enhancement may not be obvious if you are working on a single machine where no network interaction is required, but in real world applications it can make all the difference between a sluggish application that calls the server on every user action and a crisp interface with no delays.

The second important enhancement is in memory management. You might think based on what you've read and observed to date that the EntityDataCache continually accumulates data throughout its existence. If this were the case you would very quickly witness severe performance degradation. In reality the EntityDataCache is keeping track of what is stored within it and as data is no longer required is releasing it performing a self-cleansing operation at the same time. All of this is being done without the need for you to add extra code, and more importantly still it is ensuring that data integrity is being preserved at all times. Data won't be released if required by other data, nor will data that has been altered in some way without those alterations having being saved. This also relates to performance, because getting rid of unnecessary objects in memory improves performance; and vice versa, keeping large numbers of obsolete objects in memory leads to performance degradation.

We mentioned before how **EF DataSource** simplifies context management by eliminating the need to create multiple data contexts thanks to the client cache. Now we should explain what the EntityDataCache actually is and how we can further use this information to our advantage.

The EntityDataCache is essentially the context. In terms of **EF DataSource** namespaces, the cache is the **C1.Silverlight.Data.RiaServices.RiaClientCache** class, and it is in one-to-one correspondence with **DomainContext** through its **DomainContext** property. Both the cache and its underlying **DomainContext** are created for you automatically if you use **C1DataSource** component ; however, you can create them explicitly and set the **C1DataSource.ClientCache** property in code, if necessary.

To see how simple application code becomes thanks to the client-side cache, let's add functionality of saving modified data to the project in Server-Side Filtering.

1. Simply add a button to the form and add a handler code for it:

   To write code in Visual Basic

   Visual Basic

   ```
   Private Sub button1_Click(sender As System.Object, e As
   System.Windows.RoutedEventArgs)
       c1DataSource1.ClientCache.SaveChanges()
       End Sub
   ```

   To write code in C#

   C#

   ```
   private voidbutton1_Click(object sender, RoutedEventArgs e)

   {

      c1DataSource1.ClientCache.SaveChanges();

   }
   ```

2. Save, build and run the application.

- Select a category and then make some changes to the product information in the grid.
- Select another category (and possibly a third) and again makes changes to the product details in the grid.
- Click the button you added, close the application, reopen it and select the same categories as before.
- Observe how the changes you made have been saved. **EF DataSource** has provided, via the EntityDataCache, a way to alter the product details of several categories' products without the need to save those changes each time different category is selected. To achieve this effect without **EF DataSource**, you would either need to write a lot of code, or you'd need to keep the entities (the product details) from different categories in the same context. This would waste memory without releasing it, creating a memory leak. **EF DataSource** simplifies all of this for you while optimizing memory usage and performance.

Client-side caching also makes possible other important features of **EF DataSource**, such as client-side queries and, especially, live views. Live Views is a feature that allows you to replace much of the complex application coding with simple data binding, which we'll learn more about later.

Master-Detail Binding

As we've already seen in the Server-Side Filtering example, **C1DataSource** supports master-detail binding. With very large datasets, server-side filtering is by far the best solution, but with smaller datasets, client-side filtering can be just as efficient. This next scenario demonstrates client-side master-detail binding using a grid, instead of a combo box like in our previous example, to select our categories.

To implement master-detail binding, follow these steps:

1. Using the project we created to demonstrate Server-Side Filtering, add a new form with a **C1DataSource** component using the same **DomainContextTypeName** as before and create a view source based on *GetCategories*. Note that you can make this the startup form to save time when you run the project.
2. Next, add the master grid to the form and bind it to *GetCategories*:

   ```
   ItemsSource="{Binding [GetCategories],ElementName=c1DataSource1}"
   ```

   These are the same steps we followed earlier in the simple binding scenario.

3. Now, add the detail grid to the form and bind it to the *Products* of the category currently selected in the master grid. This can be done using the following binding in XAML:

   ```
   ItemsSource="{Binding SelectedItem.Products, ElementName=dataGrid1}"
   ```

   There is one more thing needed to make it work: tell RIA Services to include all products belonging to a category with categories when it fetches data using the *GetCategories* method of the domain service. If we don't do this, *GetCategories* will fetch only categories data, without their products, so our bottom grid will be empty.

4. Manually modify the domain service (default name **DomainService1.cs**) and metadata (default name **DomainService1.metadata.cs**) files generated by Visual Studio when you added the **Domain** Service class to the Web project. In **DomainService1.cs**, add the **Include** operator to the query in the *GetCategories* method:

To write code in Visual Basic

Visual Basic

```
Return Me.ObjectContext.Categories.Include("Products")
```

To write code in C#

C#

```
returnthis.ObjectContext.Categories.Include("Products");
```

5. In the **DomainService1.metadata.cs**. add the **[Include]** attribute before the corresponding property:

   To write code in Visual Basic

   Visual Basic

   ```
   <Include()>
    Public Property Products As EntityCollection(Of Product)
   ```

   To write code in C#

   C#

   ```
   [Include]

   public EntityCollection<Product> Products { get; set; }
   ```

6. Save, build and run the application. Select a category in the first grid and notice the products that belong to that category on the bottom grid. You can still edit the categories and products exactly as before.

Large Datasets: Paging

To show large amounts of data without bringing it all to the client at once, Web applications have traditionally used paging. Paging is not an ideal solution; it complicates the interface and makes it less convenient for the user, but it is preferred in some applications. For these cases, **C1DataSource** supports paging as expected, but it does so without the traditional limitations on data modification. The user can make changes in multiple pages in one session without being forced to send the changes from one page to the database before moving to the next page. That's a substantial enhancement compared to other paging implementations, such as the one in the standard RIA Services DomainDataSource.

Note: ComponentOne Entity Framework DataSource (EF DataSource) does offer a solution to the drawbacks of paging; we'll cover virtual mode later in this documentation.

To implement paging, follow these steps:

1. Using the project we created to demonstrate Master-Detail Binding, add a new form with a **C1DataSource** component using the same **DomainContextTypeName** as before. Note that you can make this the startup form to save time when you run the project.
2. Create a **ViewSource** in the **ViewSourceCollection** editor and select *GetOrders* in the **QueryName** property.
3. To enable paging, set the **PageSize** property to 10 for now, but you can choose any reasonable value for this property. It's simply determining the number of data rows that will be displayed on a page.
4. Add a **DataGrid** and bind it to O*rders* the same way we did it before:
5.
6. `ItemsSource="{Binding [GetOrders],ElementName=c1DataSource1}"`

7. Set the **AutoGenerateColumns** property of the grid to *True* in XAML.
8. To allow the user to move between pages and to show the current page and page count, add two buttons and a label using, for example, the following XAML:
9.
10. `<StackPanel Orientation="Horizontal" Grid.Row="1" Margin="10">`
11.
12. `<Button Padding="10,0,10,0" Margin="1" Content="<" Click="MoveToPrevPage"/>`
13.
14. `<TextBlock VerticalAlignment="Center" Text="Page: "/>`
15.
16. `<TextBlock VerticalAlignment="Center" x:Name="pageInfo"/>`
17.
18. `<Button Padding="10,0,10,0" Margin="1"  Content=">" Click="MoveToNextPage"/>`
19.
20. `</StackPanel>`
21. Add the following code containing a **RefreshPageInfo** handler for the **PropertyChanged** event used to show current page number and page count, and handlers for the **buttonClick** events used to move to the next and previous pages:

To write code in Visual Basic

Visual Basic

```
Imports C1.Data.DataSource

 Partial Public Class Paging
                    Inherits UserControl

    Private _view As ClientCollectionView


                    Public Sub New()
                       InitializeComponent()

        _view = c1DataSource1("GetOrders")

        RefreshPageInfo()

        AddHandler _view.PropertyChanged, AddressOf RefreshPageInfo

    End Sub

    Private Sub RefreshPageInfo()
                       pageInfo.Text = String.Format("{0} / {1}", _view.PageIndex + 1,
_view.PageCount)
                    End Sub

    Private Sub MoveToPrevPage_Click(sender As System.Object, e As
System.Windows.RoutedEventArgs) Handles MoveToPrevPage.Click
                       _view.MoveToPreviousPage()
                    End Sub

    Private Sub MoveToNextPage_Click(sender As System.Object, e As
System.Windows.RoutedEventArgs) Handles MoveToNextPage.Click
                       _view.MoveToNextPage()
                    End Sub
                 End Class
```

To write code in C#

C#

```
using C1.Data.DataSource;

namespaceTutorialsSilverlight
{
    public partial class Paging : UserControl
    {
        ClientCollectionView _view;

        public Paging()
        {
            InitializeComponent();

            _view = c1DataSource1["GetOrders"];

            RefreshPageInfo();

            _view.PropertyChanged += delegate { RefreshPageInfo(); };
        }

        private void RefreshPageInfo()
        {
            pageInfo.Text = string.Format("{0} / {1}",
                                _view.PageIndex + 1, _view.PageCount);
        }
```

```
        private void MoveToPrevPage(object sender, RoutedEventArgs e)
        {
            _view.MoveToPreviousPage();
        }

        private void MoveToNextPage(object sender, RoutedEventArgs e)
        {
            _view.MoveToNextPage();
        }
    }
}
```

22. Save, build and run the application.

    Notice there is an empty *Customer* column in the grid. This is because we only fetch *Order* objects without associating them with *Customer* objects. This can be rectified by using the **[Include]** attribute, as we did in Master-Detail Binding.

23. In the **DomainService1.cs**, add the **Include** operator to the query in the *GetOrders* method:

    To write code in Visual Basic

    Visual Basic

```
Return Me.ObjectContext.Orders.Include("Customer")
```

    To write code in C#

    C#

```
return this.ObjectContext.Orders.Include("Customer");
```

24. In the **DomainService1.metadata.cs**, add the **[Include]** attribute before the corresponding property:

To write code in Visual Basic

Visual Basic

```
<Include()>
 Public Property Customer As Customer
```

To write code in C#

C#

```
[Include]

publicCustomer Customer { get; set; }
```

25. Run the project and page through all of the *Orders*. While you are moving between pages, try changing some data in the grid. Try changing data in one page, then move to another page and try changing data there. Notice that **C1DataSource** allows you to do this without forcing you to save data to the database before leaving the page. This is an important enhancement compared with other paging implementations, including the one supported by DomainDataSource in Microsoft RIA Services (which is for Silverlight only, whereas **EF DataSource** supports this, as other features, for all three platforms: WinForms, WPF, Silverlight).

    Try to delete some orders. This is also allowed without restrictions, and moreover, the current page will automatically complete itself to keep the number of rows in the page unchanged.

    And if you add a **Save Changes** button as we did above, using the following handler, then you will be able to save changes made in multiple pages by pressing that button when you are done.

    To write code in Visual Basic

    Visual Basic

```
Private Sub btnSaveChanges_Click(sender As System.Object, e As
System.Windows.RoutedEventArgs) Handles button1.Click
    c1DataSource1.ClientCache.SaveChanges()
 End Sub
```

To write code in C#

C#

```
private voidbtnSaveChanges_Click(object sender, RoutedEventArgs e)
{
    c1DataSource1.ClientCache.SaveChanges();
}
```

All this functionality is what you would expect from a paging implementation that supports unrestricted data modification. Unfortunately, it is not easy to implement.

For example, think of all possible cases where changing data on one page interferes with what should be shown on other pages, and so on. That is why paging usually imposes severe restrictions on data modifications, if they are at all allowed. For example, MS DomainDataSource requires you to save all changes before changing pages. Fortunately, **EF DataSource** supports paging without restricting data modifications in any way.

As with many other features, unlimited modifiable paging is made possible by the client-side cache, see The Power of Client Data Cache. That also means that paging implementation in **EF DataSource** is optimized both for performance and for memory consumption. The cache provides that optimization. It keeps recently visited pages in memory, so re-visiting them is usually instantaneous. And it manages memory resources, releasing old pages when necessary, to prevent memory leaks.

Large Datasets: Virtual Mode

As mentioned in the Large Datasets: Paging topic, **ComponentOne Entity Framework DataSource** (**EF DataSource**) has an even better solution than paging for dealing with large data and large numbers of rows.

What if using large datasets with thousands, or even millions, of rows was no longer a problem? What if you could use the same controls to display massive datasets as you would for smaller datasets, with no paging, no code changes, and all by setting one Boolean property? With **EF DataSource** you can, thanks to the magical **VirtualMode** property.

To implement virtual mode, follow these steps:

1. Using the project we created to demonstrate Large Datasets: Paging, add a new form with a **C1DataSource** component using the same **DomaintContextTypeName** as before. Note that you can make this the startup form to save time when you run the project.

2. Create a **ViewSource** in the **ViewSourceCollection** editor. Use the largest table in our sample database: *Order_Details*.
3. Set the **VirtualMode** property to *Managed*.Another possible value is *Unmanaged*, but that is an advanced option that should be used with caution and only when necessary. The *Managed* option means that getting data from the server is managed by a grid control. With the *Managed* option, EF DataSource supports the ComponentOne grid controls: C1FlexGrid and C1DataGrid. It is optimized for performance with those specific grid controls. The *Unmanaged* option means that virtual mode is not driven by a specific control, will work with any bound controls but is subject to some limitations described here.
4. Add a grid to the designer and bind it to *GetOrder_Details* the same way we did before:
5.
6. `ItemsSource="{Binding [GetOrder_Details],ElementName=c1DataSource1}"`
7. Now, since we selected the *Managed* option, we need to specify which grid control is driving the data. **EF DataSource** defines an attached property **C1DataSource.ControlHandler** that is an object having properties affecting the control's behavior when it is bound to a **C1DataSource**. There is a boolean **VirtualMode** property in **C1DataSource.ControlHandler** that marks the control as the main, "driving" control in *Managed* virtual mode. Add the following inside the DataGrid markup in XAML (the added markup is shaded):
8.
9.
```
<sdk:DataGrid AutoGenerateColumns="True" Name="dataGrid1"
                ItemsSource="{Binding
[GetOrder_Details],ElementName=c1DataSource1}">
                    <c1:C1DataSource.ControHandler>
                           <c1:ControlHandler VirtualMode="True"/>
                    </c1:C1DataSource.ControHandler>
               </DataGrid>
```
10. Save, build and run the application. You'll see nothing earth-shattering, just a grid you can navigate, scroll and modify as necessary. It looks and behaves like any conventional data grid, which is exactly the point. You can use large datasets without the drawbacks of paging and without code. And as an added benefit, you can use any GUI control you like so long as it has a **DataSource** property. This example uses a relatively modest-sized dataset, but **C1DataSource** running in virtual mode would be just as responsive with a much larger dataset; it does not depend on the number of rows. To further prove the point, look at the **OrdersDemo** sample installed with this product. The sample uses a larger database, with roughly 65,000 additional rows, but the responsiveness is the same as our example here. Again, it does not depend on the number of rows in the dataset.

How does this magic work? It works much like paging, only hiding its inner workings from the GUI controls, with paging occurring under the hood. The GUI controls see the data as if it is fetched to the client and ready for them. When GUI controls request data, **C1DataSource**, or **ClientViewSource** if it is used in code without a **C1DataSource** control, first checks whether it can serve the data from memory, from the same client-side cache it uses for all features. If it can't find it in memory, it transparently fetches the required data from the server. As with other features using the client-side cache, **C1DataSource** does not store the fetched data indefinitely, which would be a memory leak. It knows which parts of the data are needed for serving the GUI controls and which parts should be kept because they are modified or related to modified parts, and so on. It releases old, unnecessary parts of data as needed. No code is required and any GUI controls can be used.

Automatic Lookup Columns in Grids

A common scenario in data binding is for data classes to contain references to other data classes. For example, a **Product** object may contain references to **Category** objects.

In ADO.NET, the references usually appear as foreign keys that map in other tables (e.g., **Product.CategoryID**).

In the Entity Framework and RIA Services, you still get the key columns, but you also get the actual objects. So you have **Product.CategoryID** (usually an integer) and **Product.Category** (an actual Category object).

Displaying foreign keys in a grid is not very useful, because it is unlikely that users will remember that category 12 is "Dairy Products". Allowing users to edit these keys would be even worse. A common way to work around this problem is to remove the related entity columns from any bound grids and, optionally, to replace them with custom columns that use combo boxes for editing the values, so called lookups. The combo boxes have to be bound to the related tables and have their properties set so they display relevant values (e.g., **Category.Name**) and so they are synchronized with the values being displayed on the grid. This is not terribly hard to do, but it is a tedious and error-prone task that makes projects harder to create and maintain.

C1DataSource can do this tedious work for the developer; it can automatically change related entity columns so that they show combo box lookups. It can do this for several types of data grids, those that it supports. Currently supported Silverlight grids are: C1FlexGrid and C1DataGrid. Here we will show how to do this for **C1FlexGrid**.

C1DataSource provides an *attached* property called **ControlHandler**. If you place a **C1FlexGrid** control on a designer surface that contains a **C1DataSource**, you can set an additional **C1DataSource.ControlHandler** property on the grid. A ControlHandler is an object containing (at present) a single boolean property **AutoLookup**. This property set to *True* causes the **C1DataSource** to configure grid columns that contain references to other entities so that they show lookup combos.

To see how it works, follow these steps:

1. Using the project used in Simple Binding, add a new form with a **C1DataSource** control using the same **ObjectContextType** as before.
2. Create a **ViewSource** in the **ViewSourceCollection** editor, entering *Products* as the **EntitySetName**.
3. Add a **C1FlexGrid** to the designer and bind it to *Products* as we did it before:

   ```
   ItemsSource="{Binding[Products], ElementName=c1DataSource1}"
   ```

4. Save, build and run the application. It will look like this:

Product ID ▲	Product Name	Category
5	Chef Anton's Gumbo Mix	Category : 2
9	Mishi Kobe Niku	Category : 6
17	Alice Mutton	Category : 6
24	Guaraná Fantástica	Category : 1
28	Rössle Sauerkraut	Category : 7
29	Thüringer Rostbratwurst	Category : 6
42	Singaporean Hokkien Fried Me	Category : 5
53	Perth Pasties	Category : 6

As you can see, the **Category** column is not very useful. You could remove it or customize the grid by writing some code to create a new column, but there's an easier way.

5. Add the following inside the **C1FlexGrid** markup in XAML (the added markup is shaded):
6.
7.
```
        <c1:C1FlexGrid AutoGenerateColumns="True" Name="dataGrid1"
```
8.
```
             ItemsSource="{Binding
GetProducts],ElementName=c1DataSource1}" >
```
9.
```
                <c1:C1DataSource.ControHandler>
```
10.
```
                      <c1:ControlHandler AutoLookup="True"/>
```
11.
```
                </c1:C1DataSource.ControHandler>
```
12.
```
        </c1:C1FlexGrid>
```
13. Run the project again and look at the **Category** column. Notice that now the grid shows the category name instead of the cryptic strings. Also notice that you can edit the product's category by picking from a drop-down list, complete with auto search functionality.

Product ID ▲	Product Name	Category
5	Chef Anton's Gumbo Mix	Condiments
9	Mishi Kobe Niku	Meat/Poultry
17	Alice Mutton	Meat/Poultry
24	Guaraná Fantástica	Beverages
28	Rössle Sauerkraut	Beverages
29	Thüringer Rostbratwurst	Condiments
42	Singaporean Hokkien Fried Me	Confections
53	Perth Pasties	Dairy Products
		Grains/Cereals

14. Finally, click the column header to sort the grid by **Category** and notice that the sorting is performed based on the value displayed on the grid. This is what anyone would expect, but surprisingly it is not easy to achieve using regular data binding.

The string value (name) shown by the combo box is determined following these rules:

1. If the entity class overrides the **ToString** method, then the string representation of the entity is obtained using the overridden **ToString** method. This should return a string that uniquely represents the entity. For example, it could be the content of a **CompanyName** column, or a combination of **FirstName**, **LastName**, and **EmployeeID**. This is the preferred method because it is simple, flexible, and easy to implement using partial classes (so your implementation will not be affected if the entity model is regenerated).
2. If the entity class does not override the **ToString** method, but one of its properties has the **DefaultProperty** attribute, then that property is used as the string representation of the entity.
3. If the entity class does not override the **ToString** method and has no properties marked with the **DefaultProperty** attribute, then the first column that contains the string "Name" or "Description" in its name will be used as a string representation for the entities.
4. If none of the above applies, then no lookup is created for the given entity type.

Customizing View

In many situations, you may want to use custom views that do not correspond directly to the tables and views provided by the database. LINQ is the perfect tool for these situations. It allows you to transform raw data using your language of choice using query statements that are flexible, concise, and efficient. **ComponentOne Entity Framework DataSource (EF DataSource)** makes LINQ even more powerful by making LINQ query statements *live*. That's why its LINQ implementation is called LiveLinq. We will show the power of LiveLinq in Live Views. For now suffice it to say that you can transform your view to shape it whatever way you want using LINQ operators without losing full updatability and bindability.

Let's try, for example, one of the LINQ operators: **Select** (also called projection), to customize the fields (properties) of our view.

To customize a view, follow these steps:

1. Using the project we created to demonstrate Large Datasets: Paging, add a new form with a **C1DataSource** component using the same **DomainContextTypeName** as before. Note that you can make this the startup form to save time when you run the project.
2. Create a **ViewSource** in the **ViewSourceCollection** editor. Use the *GetProducts* table in our sample database.
3. Add a **DataGrid** to the page designer and, as before, set its **AutoGenerateColumns** property to *True*. But this time we won't bind the grid to **C1DataSource** in XAML. Instead, we will bind it in code to implement a custom view.
4. Create a custom live view and bind the grid to that view with the following code:

To write code in Visual Basic

Visual Basic

```
dataGrid1.ItemsSource = _
    (From p In c1DataSource1("GetProducts").AsLive(Of Product)()
     Select New With
     {
        p.ProductID,
        p.ProductName,
        p.CategoryID,
        p.Category.CategoryName,
        p.SupplierID,
        .Supplier = p.Supplier.CompanyName,
        p.UnitPrice,
        p.QuantityPerUnit,
        p.UnitsInStock,
        p.UnitsOnOrder
     }).AsDynamic()
```

To write code in C#

C#

```
dataGrid1.ItemsSource =

   (from p in c1DataSource1["GetProducts"].AsLive<Web.Product>()

    select new

    {

        p.ProductID,

         p.ProductName,

        p.CategoryID,

        CategoryName = p.Category.CategoryName,

        p.UnitPrice,
```

```
        p.QuantityPerUnit,

        p.UnitsInStock,

        p.UnitsOnOrder

    }).AsDynamic();
```

Here c1DataSource1["GetProducts"] is a ClientCollectionView object. It is the view that is created by the view source that we set up in the designer (if you need to access the view source itself in code, it is also available, as c1DataSource.ViewSources["GetProducts"]). The **AsLive<Web.Product>()** extension method call is needed to specify the item type of the view (Web.Product is the class generated for the *Product* entity in DomainService) so **LiveLinq** operators can be applied to it. The result, c1DataSource1["GetProducts"].AsLive<Web.Product>(), is a View<Web.Product>. The C1.LiveLinq.LiveViews.View is the main class of LiveLinq, used for client-side live views. LINQ operators applied to live views preserve their updatability and bindability. They are the same usual LINQ operators, but the fact that they are applied to a live view gives them these additional features that are critically important for data binding applications.

Note 1: AsDynamic() must be applied to this view because its result selector (the select new... code) uses an anonymous class. This is a minor LiveLinq limitation, only for anonymous classes. If you forget to add AsDynamic() to such view, you will be reminded by an exception.

Note 2: At this point we will need to add the following two attributes to the AssemblyInfo file in our project:

To write code in Visual Basic

Visual Basic

```
<Assembly: InternalsVisibleTo("System.Core, PublicKey=" +
C1LiveLinqInfo.PublicKey_System_Core)>

<Assembly: InternalsVisibleTo("C1.Silverlight.LiveLinq, PublicKey=" +
C1LiveLinqInfo.PublicKey_C1_Silverlight_LiveLinq)>
```

To write code in C#

C#

```
[assembly: InternalsVisibleTo("System.Core, PublicKey=" +
C1LiveLinqInfo.PublicKey_System_Core)]

[assembly: InternalsVisibleTo("C1.Silverlight.LiveLinq, PublicKey=" +
C1LiveLinqInfo.PublicKey_C1_Silverlight_LiveLinq)]
```

These attributes give LiveLinq access to non-public members of our assembly, If we don't add them, LiveLinq will remind us at run time by throwing an exception. This is necessary only in Silverlight and only if we use anonymous or non-public types in LiveLinq queries.

5. Since we want to show the value of p.Category.CategoryName, we need to make sure that for every product entity, its corresponding *Category* entity is fetched to the client. This can be done either by fetching all *Category* entities to the client in a separate query or by telling RIA Services that we want it to fetch *Category* with every *Product*. Let's choose the second option and use the **Include** attribute in the **DomainService1.cs** and **DomainService1.metadata.cs** files as we already did before.

   In **DomainService1.cs**, add the **Include** operator to the query in the *GetProducts* method:

   To write code in Visual Basic

   Visual Basic

```
Return Me.ObjectContext.Products.Include("Category")
```

   To write code in C#

   C#

```
return this.ObjectContext.Products.Include("Category");
```

6. In **DomainService1.metadata.cs**, add the **[Include]** attribute before the corresponding property:

   To write code in Visual Basic

Visual Basic

```
<Include()>
Public Property Category As Category
```

To write code in C#

C#

```
[Include]

publicCategory Category { get; set; }
```

7. Save, build and run the application. The grid now shows the columns we defined in the 'select' clause of our **LiveLinq** view. Note also that all columns are modifiable. It may not seem like a big deal; however, it is an important feature that would be difficult to implement on your own for this customized view, especially when adding and deleting rows, which, as you can see here, is also supported.
8. To add or delete rows, add two buttons to the page executing the following code:

To write code in Visual Basic

Visual Basic

```
 Private Sub addItem_Click(sender As System.Object, e As
System.Windows.RoutedEventArgs) Handles addItem.Click
                    Dim newItem = CType(dataGrid1.ItemsSource,
System.ComponentModel.IEditableCollectionView).AddNew()
                    dataGrid1.ScrollIntoView(dataGrid1.Rows.Count - 1, 0)
                End Sub

 Private Sub deleteItem_Click(sender As System.Object, e As
System.Windows.RoutedEventArgs) Handles deleteItem.Click
                    Dim item = CType(dataGrid1.ItemsSource,
System.ComponentModel.ICollectionView).CurrentItem
                    If item IsNot Nothing Then
                       CType(dataGrid1.ItemsSource,
System.ComponentModel.IEditableCollectionView).Remove(item)
                    End If
```

```
End Sub
```

To write code in C#

C#

```
private voidaddItem(object sender, RoutedEventArgs e)

{

  objectnewItem =
                    ((System.ComponentModel.IEditableCollectionView)dataGrid1.ItemsSource).
AddNew();

  dataGrid1.ScrollIntoView(newItem, null);

}


private voiddeleteItem(object sender, RoutedEventArgs e)

{

  varitem =
                    ((System.ComponentModel.ICollectionView)dataGrid1.ItemsSource).CurrentI
tem;

  if(item != null)

    ((System.ComponentModel.IEditableCollectionView)dataGrid1.ItemsSource).Remove(item
);

}
```

This is needed only because Microsoft DataGrid does not provide a built-in interface for adding new rows.

As we just saw, LiveLinq operators preserve updatability. But we also said that they preserve bindability. Bindability is achieved because the views are always 'live'; they aren't simple

snapshots of static data. The views "feel" and automatically reflect changes in their source data. We will see this remarkable feature in action in the Live Views topic.

Working with Data Sources in Code

Up to this point, we have been setting up data sources directly on the designer surface with very little code. **ComponentOne Entity Framework DataSource (EF DataSource)** has made it very easy, but sometimes you want or need to do everything in code. **EF DataSource** makes this possible as well. Everything we did previously can be done at run time in code.

An obvious way to go about this would be to use the **ClientViewSource** object that we have, in effect, been setting up in the designer as elements of the **ViewSourceCollection** of a **C1DataSource**, given that it can be created on its own without a **C1DataSource**. We could, however, take a step further back and use a lower level class ClientView<T>. This would provide full control over loading data from the server and, since it is derived from **C1.LiveLinq.LiveViews.View<T>**, we can apply any LiveLinq operators to it. The ability to bind this to any GUI control whose datasource can be set to a **View<T>** also means that we'll end up with a fully editable view of our data.

Server-side filtering is, perhaps, the most common operation, because no one usually wants entire database tables brought to the client unrestricted. Earlier we saw how **EF DataSource** made it simple to perform without code, but now we'll try it in run-time code.

To begin using **EF DataSource** in run-time code (without **C1DataSource**), add a few lines to our program's main class to create a global client-side data cache. When we use **C1DataSource**, it is created for us behind the scenes. Now we create it explicitly using the following code:

To write code in Visual Basic

Visual Basic	Copy Code
Partial Public Class App Inherits Application Public Sub New() InitializeComponent() End Sub Public Shared ClientCache As RiaClientCache Private Sub Application_Startup(ByVal o As Object, ByVal e As StartupEventArgs) Handles Me.Startup	

```
            Me.RootVisual = New MainPage()

        ClientCache = RiaClientCache.GetDefault(GetType(Web.DomainService1))
            End Sub
```

To write code in C#

C#	Copy Code

```
usingC1.Silverlight.Data.RiaServices;



public partial classApp : Application

{

    public static RiaClientCache ClientCache;

    public static Web.DomainService1 DomainContext;



    private void Application_Startup(object sender, StartupEventArgs )

    {

        this.RootVisual = new MainPage();



        DomainContext = new Web.DomainService1();

        ClientCache = new RiaClientCache(DomainContext);

    }
```

This code creates a single application-wide (static) **DomainContext** and associates it with **RiaClientCache**. As noted previously in The Power of Client Data Cache topic, the ability to have a

single context (and cache) for the entire application is a great simplification made possible by **EF DataSource**.

To perform server-side filtering in run-time code, follow these steps:

1. Add a new form using the project created to demonstrate Customizing View.
2. Add a grid (**dataGrid1** and set its **AutoGenerateColumns** property to *True*), a combo box (**comboBox1**), and a button (**btnSaveChanges**) to the form.
3. Add the following code to the user control class:

   To write code in Visual Basic

Visual Basic	Copy Code

```
Imports C1.Data
Imports C1.Silverlight.Data.RiaServices

Partial Public Class DataSourcesInCode
    Inherits UserControl

    Private _scope As RiaClientScope

    Public Sub New()
    InitializeComponent()

        _scope = App.ClientCache.CreateScope()

        Dim viewCategories As ClientView(Of Category) = _scope.GetItems(Of
Category)("GetCategories")

        comboBox1.DisplayMemberPath = "CategoryName"
                    comboBox1.ItemsSource = viewCategories
                End Sub

    Private Sub BindGrid(categoryID As Integer)
                    dataGrid1.ItemsSource =
                        (From p In _scope.GetItems(Of Product)("GetProducts").AsFiltered(Function(p
As Product) p.CategoryID.Value = categoryID)
                         Select New With
                         {
                            p.ProductID,
```

```
                                p.ProductName,
                                p.CategoryID,
                                p.Category.CategoryName,
                                p.SupplierID,
                                .Supplier = p.Supplier.CompanyName,
                                p.UnitPrice,
                                p.QuantityPerUnit,
                                p.UnitsInStock,
                                p.UnitsOnOrder
                            }).AsDynamic()
                    End Sub


                    Private Sub comboBox1_SelectionChanged(sender As System.Object, e As
System.Windows.Controls.SelectionChangedEventArgs) Handles comboBox1.SelectionChanged
                        If comboBox1.SelectedValue IsNot Nothing Then
                            BindGrid(CType(comboBox1.SelectedValue, Category).CategoryID)
                        End If
                    End Sub

    Private Sub btnSaveChanges_Click(sender As System.Object, e As
System.Windows.RoutedEventArgs) Handles btnSaveChanges.Click
                        App.ClientCache.SaveChanges()
                    End Sub
                End Class
```

To write code in C#

C#	Copy Code

```
using C1.Silverlight.Data.RiaServices;

using C1.Data;
```

```
public partial classDataSourcesInCode : UserControl

{

    private RiaClientScope _scope;



    public DataSourcesInCode()

    {

        InitializeComponent();



        _scope = App.ClientCache.CreateScope();



        ClientView<Web.Category> viewCategories =
                                _scope.GetItems<Web.Category>("GetCategories");



        comboBox1.DisplayMemberPath = "CategoryName";

        comboBox1.ItemsSource = viewCategories;

    }



    private void comboBox1_SelectionChanged(object sender,
                                SelectionChangedEventArgs e)

    {

        if (comboBox1.SelectedValue != null)

            BindGrid(((Web.Category)comboBox1.SelectedValue).CategoryID);
```

```
}

private void BindGrid(int categoryID)
{
    dataGrid1.ItemsSource =
        (from p in _scope.GetItems<Web.Product>("GetProducts")
                        .AsFiltered(p => p.CategoryID == categoryID)
         select new
         {
             p.ProductID,
             p.ProductName,
             p.CategoryID,
             CategoryName = p.Category.CategoryName,
             p.UnitPrice,
             p.QuantityPerUnit,
             p.UnitsInStock,
             p.UnitsOnOrder
         }).AsDynamic();
}

private void btnSaveChanges_Click(object sender, RoutedEventArgs e)
{
```

```
        App.ClientCache.SaveChanges();

    }

}
```

4. Save, build and run your application. You should see similar results to those you saw in the Server-Side Filtering example, the only difference being that this time we've implemented it all in code.

Let's take a closer look at some of the code we've just written.

The private field **_scope** is the form's gateway to the global data cache. It is a pattern we recommend you follow in all forms where you do not employ a **C1DataSource** component directly, as that does this for you automatically. It ensures that the entities the form needs stay in the cache while the form is alive, and that they are automatically released when all forms (scopes) holding them are released.

Creating a view showing all categories for the combo box is simple:

To write code in Visual Basic

Visual Basic	Copy Code
Dim viewCategories As ClientView(Of Category) = _scope.GetItems(Of Category)("GetCategories")	

To write code in C#

C#	Copy Code
ClientView<Web.Category> viewCategories = _scope.GetItems<Web.Category>("GetCategories");	

To create the view to which the grid is bound that only provides those products associated with the chosen category in the combo box required one additional operator; AsFiltered(<predicate>).

To write code in Visual Basic

Visual Basic	Copy Code

```
From p In _scope.GetItems(Of Product)("GetProducts").AsFiltered(Function(p As Product)
p.CategoryID.Value = categoryID)
```

To write code in C#

C#	Copy Code

```
from p in _scope.GetItems<Web.Product>("GetProducts")
    .AsFiltered(p => p.CategoryID == categoryID)
```

Note that when this query is executed, the result does not necessarily require a round trip to the server to retrieve the products requested. The cache is examined first to see if it already contains the requested data, either because the required data has already been requested once before within this form or from another form in the application. Or, possibly a completely separate query run elsewhere in the application had requested that all products be returned, so the cache would already have all the product data. Again, this is a fundamental strength of **EF DataSource**. By providing your application with a global cache of data, its performance is continually improved throughout its lifetime.

Here we chose to create a new view, and bind the grid to it, every time the user selects a new category in the combo box (see the combo box's **SelectedValueChanged** event). However, we could have avoided the need to create new views all the time and instead created one single view using a special **BindFilterKey**, which we'll learn more about in the Simplifying MVVM topic.

So, in summary, we replicated in code what we did on the design surface with **C1DataSource** in Server-Side Filtering. We have even thrown in a little extra; we customized the fields shown in the grid columns as we did in Customizing View by adding a **Select** to our LiveLinq statement:

To write code in Visual Basic

Visual Basic	Copy Code

```
Select New With
{
    p.ProductID,
    p.ProductName,
    p.CategoryID,
    p.Category.CategoryName,
    p.SupplierID,
```

```
    .Supplier = p.Supplier.CompanyName,
    p.UnitPrice,
    p.QuantityPerUnit,
    p.UnitsInStock,
    p.UnitsOnOrder
}
```

To write code in C#

C#	Copy Code

```
select new

{

    p.ProductID,

    p.ProductName,

    p.CategoryID,

    CategoryName = p.Category.CategoryName,

    p.UnitPrice,

    p.QuantityPerUnit,

    p.UnitsInStock,

    p.UnitsOnOrder

};
```

Had we just wanted the raw product data returned from the table without any special formatting, we could have simply said;

```
select p;
```

Live Views

Live views is a powerful feature, so let's take a little more time to see what live views can do. Live views are designed to make data binding more powerful, so powerful, in fact, that you can develop virtually entire applications with just LINQ (in its LiveLinq form) and data binding.

Live views, called **LiveLinq**, a part of **ComponentOne Entity Framework DataSource** (**EF DataSource**), is a client-side, in-memory feature that is applicable to Entity Framework and RIA Services data, as we saw above, but to any observable collections in memory, including XML (LINQ to XML object model) and ADO.NET DataSets.

So, for example, you can use live views over RIA Services data and some other data (for example, XML retrieved from some web service) to integrate that data and to provide easy full-featured data binding to the integrated data. This is a powerful tool for building applications that get data from various sources. For now we're going to concentrate on how we can use it with the Entity Framework, but if you'd like to explore **LiveLinq** in greater depth, see the ComponentOne LiveLinq documentation.

In fact, we already saw an example of such customization in Customizing View. But there we only changed the properties (fields) of the view and only applied one LINQ operator, **Select**. Let's apply some more LINQ operators to transform the view. What we do here will be similar to what we did in Customizing View, but instead of using **C1DataSource**, we'll be doing everything in code.

To use live views, follow these steps:

1. Add a new form using the project created to demonstrate Working with Data Sources in Code, and add a data grid, **dataGridView1**, to the form.
2. In the form's Load event, add the following code:

   To write code in Visual Basic

Visual Basic	Copy Code
Private _scope As RiaClientScope = App.ClientCache.CreateScope()	

   To write code in C#

C#	Copy Code
privateRiaClientScope _scope = App.ClientCache.CreateScope();	

Here we are following the pattern recommended in Working with Data Sources in Code and creating a field in the user control class.

3. Getting *Products* data from the scope, we create a live view and bind the grid to that view in the user control's constructor:

To write code in Visual Basic

Visual Basic	Copy Code

```
_viewProducts =
    (From p In _scope.GetItems(Of Product)("GetProducts")
     Where Not p.Discontinued And p.UnitPrice >= 30
     Order By p.UnitPrice
     Select New With
     {
        p.ProductID,
        p.ProductName,
        p.CategoryID,
        p.Category.CategoryName,
        p.SupplierID,
        .Supplier = p.Supplier.CompanyName,
        p.UnitPrice,
        p.QuantityPerUnit,
        p.UnitsInStock,
        p.UnitsOnOrder
     }).AsDynamic()
```

To write code in C#

C#	Copy Code

```
_viewProducts =

   (from p in _scope.GetItems<Web.Product>("GetProducts")
```

```
    where !p.Discontinued && p.UnitPrice >= 30

    orderby p.UnitPrice

    select new

    {

        ProductID = p.ProductID,

        ProductName = p.ProductName,

        CategoryID = p.CategoryID,

        CategoryName = p.Category.CategoryName,

        SupplierID = p.SupplierID,

        UnitPrice = p.UnitPrice,

        QuantityPerUnit = p.QuantityPerUnit,

        UnitsInStock = p.UnitsInStock,

        UnitsOnOrder = p.UnitsOnOrder

    }).AsDynamic();

dataGrid1.ItemsSource = _viewProducts;
```

In this example, we applied several **LiveLinq** operators: *Where*, *OrderBy*, and *Select*. We defined our view as containing products that aren't discontinued and have a unit price of at least 30, and we sorted our view by unit price.

We chose to store the view in a private field **_viewProducts** here:

To write code in Visual Basic

Visual Basic	Copy Code

```
Private _viewProducts As View(Of Object)
```

To write code in C#

C#	Copy Code

```
private View<dynamic> _viewProducts;
```

That is only because we will need it later. If we did not, we could use a local variable for the view.

4. Now run the project. You see that the grid shows the filtered set of products, "expensive" products that aren't discontinued, in the order that we specified with the columns that we specified. Note also that all columns are modifiable, and you can even add and delete rows in the grid. Also note that you can sort the grid at run time by clicking column headers.

But live views offer even more great features. Standard LINQ to Objects produces snapshots of the data that cannot reflect changes in the source data, except some simple property changes, and even then under the strict proviso that you don't utilize a custom **Select** in your LINQ statement. Live Views, on the other hand, provide dynamic 'live' views that automatically reflect changes in their source data. As such, they simplify application development because you can, in most cases, rely on data binding to automate 'live' changes in views without the need to synchronize the changes in different parts of the application with code.

To see another example of how live views automatically synchronize themselves with changes in underlying data, follow these steps:

1. Add a live view member of the user control class:

   To write code in Visual Basic

Visual Basic	Copy Code

```
Private _seafoodProductsView As ClientView(Of Product)
```

To write code in C#

C#	Copy Code
private ClientView<Web.Product> _seafoodProductsView;	

2. Add the following code to the form's constructor:

To write code in Visual Basic

Visual Basic	Copy Code
_seafoodProductsView = _scope.GetItems(Of Product)("GetProducts").AsFiltered(Function(p) p.CategoryID.Value = 8)	

To write code in C#

C#	Copy Code
_seafoodProductsView = _scope.GetItems<Web.Product>("GetProducts") .AsFiltered(p => p.CategoryID == 8);	

3. Add two buttons, named **btnRaise** and **btnCut**, to the form and add the following handlers to the form's code :

To write code in Visual Basic

Visual Basic	Copy Code
Private Sub raiseSeafood_Click(sender As System.Object, e As System.Windows.RoutedEventArgs) For Each p In _seafoodProductsView	

```
            p.UnitPrice *= 1.2
        Next
    End Sub

    Private Sub cutSeafood_Click(sender As System.Object, e As System.Windows.RoutedEventArgs)
        For Each p In _seafoodProductsView
            p.UnitPrice /= 1.2
        Next
    End Sub
```

To write code in C#

C#	Copy Code

```
private voidraiseSeafood(object sender, RoutedEventArgs e)

{

    foreach (var p in _seafoodProductsView)

        p.UnitPrice *= 1.2m;

}



private voidcutSeafood(object sender, RoutedEventArgs e)

{

    foreach (varp in _seafoodProductsView)

        p.UnitPrice /= 1.2m;

}
```

4. Save, build and run the application. As you press the buttons, notice how seafood products appear in the grid because their unit price is now either greater than or equal to **30** or how

they disappear when their unit price falls below **30**. All of this happens automatically. You did not have to write any special code to refresh or synchronize the grid

To see how live views can make almost any GUI-related code easier to write (and less error-prone), let's add a text block to the form that shows the current row count. Without **EF DataSource**, this would usually be done in a method counting the rows, and that method would have to be called in every place in the code where that count can change. In this example, there would be three such places: on initial load, and in the two methods called when the buttons are pressed, raiseSeafood and cutSeafood. So it is not very easy to synchronize display with changing data even in this simple example, not to speak of a real application. Live views make all this synchronization code unnecessary. We already saw how it works for a view containing filtering, ordering, and projection (Where, OrderBy, and Select). It works as well for views containing aggregation operations, such as Count. A little difference here is that regular LINQ Count returns a single number, to which you can't bind a control, so **EF DataSource** provides a special operation LiveCount that returns a live view instead of a single number, so you can use it in data binding. We can create this binding with a single line of code:

To write code in Visual Basic

Visual Basic	Copy Code
textBlockCount.SetBinding(TextBlock.TextProperty, New Binding("Value") With {.Source = _viewProducts.LiveCount()})	

To write code in C#

C#	Copy Code
textBlockCount.SetBinding(TextBlock.TextProperty, newBinding("Value") { Source = _viewProducts.LiveCount() });	

Simplifying MVVM

The Model-View-View-Model (**MVVM**) pattern is gaining in popularity as developers realize the benefits it gives their applications, making them easier to maintain and test and, particularly in the case of WPF and Silverlight applications, allowing a much clearer division of labor between the designer of the UI and the creator of the code that makes it work. However, without effective tools

to aid program development based on **MVVM** patterns, programmers can actually find themselves working harder because of the need to implement an extra layer (the View Model) and ensure that data is properly synchronized between that and the Model. This extra burden isn't onerous when you have a relatively small application with just a few simple collections (and best practice with MVVM is to use **ObservableCollection** as the datasource), but the bigger it becomes and the more collections it spawns, the worse it becomes. **ComponentOne Entity Framework DataSource** (**EF DataSource**) can ease the burden.

EF DataSource lets you use live views as your view model. Just create live views over your model collections and use them as your view model. Live views are synchronized automatically with their sources (model), so you don't need *any* synchronization code - it's all automatic. And live views are much easier to create than to write your own view model classes using **ObservableCollection** and filling those collections manually in code. You have all the power of LINQ at your disposal to reshape model data into live views. So, not only does synchronization code disappear, but the code creating view models is dramatically simplified.

To demonstrate how easy it is to follow the **MVVM** pattern using live views, let's create a form combining all features from the last two examples: **Category-Products** master-detail from Working with Data Sources in Code and reshaping/filtering/ordering of data from Live Views. It will be a **Category-Products** view, showing non-discontinued products whose unit price is at least **30**, ordered by unit price, and displaying a customized set of product fields in a master-detail form where the user can select a category to show the products of that category. We'll follow the **MVVM** pattern. The form (view), called **CategoryProductsView**, only hosts GUI controls (a combo box and a grid) and does not have any code except what is used to set the data source to the view model.

All logic is in the view model class, separate from the GUI. More exactly, there are three classes: **CategoryViewModel**, **ProductViewModel**, and **CategoryProductsViewModel**. The first two are simple classes defining properties with no additional code:

To write code in Visual Basic

Visual Basic	Copy Code

Public Class ProductViewModel
  Public Overridable Property ProductID As Integer
  Public Overridable Property ProductName As String
  Public Overridable Property CategoryID As Integer?
  Public Overridable Property CategoryName As String
  Public Overridable Property SupplierID As Integer?
  Public Overridable Property SupplierName As String
  Public Overridable Property UnitPrice As Decimal?	

```
    Public Overridable Property QuantityPerUnit As String
    Public Overridable Property UnitsInStock As Short?
    Public Overridable Property UnitsOnOrder As Short?
End Class
```

To write code in C#

C#	Copy Code

```
public classCategoryViewModel
{
    public virtual int CategoryID { get; set; }
    public virtual string CategoryName { get; set; }
}


public classProductViewModel
{
    public virtual int ProductID { get; set; }
    public virtual string ProductName { get; set; }
    public virtual int? CategoryID { get; set; }
    public virtual string CategoryName { get; set; }
    public virtual decimal? UnitPrice { get; set; }
    public virtual string QuantityPerUnit { get; set; }
    public virtual short? UnitsInStock { get; set; }
    public virtual short? UnitsOnOrder { get; set; }
```

```
}
```

And here is the code for the **CategoryProductsViewModel** class:

To write code in Visual Basic

Visual Basic	Copy Code

```
Public Class CategoryProductsViewModel

    Private _scope As RiaClientScope

    Private _categories As ICollectionView
     Public Property Categories As ICollectionView
        Get
            Return _categories
        End Get
        Private Set(value As ICollectionView)
            _categories = value
        End Set
     End Property

    Private _products As ICollectionView
     Public Property Products As ICollectionView
        Get
            Return _products
        End Get
        Private Set(value As ICollectionView)
            _products = value
        End Set
```

```
    End Property

   Public Sub New()
       _scope = App.ClientCache.CreateScope()

      Categories =
          From c In _scope.GetItems(Of Category)("GetCategories")
          Select New CategoryViewModel With
          {
             .CategoryID = c.CategoryID,
             .CategoryName = c.CategoryName
          }

      Products =
          From p In _scope.GetItems(Of Product)("GetProducts").AsFilteredBound(Function(p)
p.CategoryID.Value).BindFilterKey(Categories, "CurrentItem.CategoryID")
          Select New ProductViewModel With
          {
             .ProductID = p.ProductID,
             .ProductName = p.ProductName,
             .CategoryID = p.CategoryID,
             .CategoryName = p.Category.CategoryName,
             .SupplierID = p.SupplierID,
             .SupplierName = p.Supplier.CompanyName,
             .UnitPrice = p.UnitPrice,
             .QuantityPerUnit = p.QuantityPerUnit,
             .UnitsInStock = p.UnitsInStock,
             .UnitsOnOrder = p.UnitsOnOrder
          }

   End Sub
 End Class
```

To write code in C#

C#	Copy Code

```
public class CategoryProductsViewModel
  {
      private C1.Silverlight.Data.RiaServices.RiaClientScope _scope;


      public System.ComponentModel.ICollectionView Categories { get; private set; }
      public System.ComponentModel.ICollectionView Products { get; private set; }


      public CategoryProductsViewModel()
      {
          if (App.ClientCache == null)
              return;


          _scope = App.ClientCache.CreateScope();


          Categories =
              from c in _scope.GetItems<Web.Category>("GetCategories")
              select new CategoryViewModel()
              {
                  CategoryID = c.CategoryID,
                  CategoryName = c.CategoryName
              };
```

```
        Products =

            from p in _scope.GetItems<Web.Product>("GetProducts")
           .AsFilteredBound(p => p.CategoryID)
           .BindFilterKey(Categories, "CurrentItem.CategoryID")

            select new ProductViewModel()

            {

                ProductID = p.ProductID,

                ProductName = p.ProductName,

                CategoryID = p.CategoryID,

                CategoryName = p.Category.CategoryName,

                UnitPrice = p.UnitPrice,

                QuantityPerUnit = p.QuantityPerUnit,

                UnitsInStock = p.UnitsInStock,

                UnitsOnOrder = p.UnitsOnOrder

            };

    }

}
```

Basically, it contains just two LiveLinq statements, nothing more.

Similar to what we saw in Working with Data Sources in Code, using **AsFilteredBound**() gives us server-side filtering. We also connected the filter key, which is the **Product.CategoryID** property, to the **CategoryID** selected by the user using a combo box event. Here we can't do this because we must keep our code independent of the GUI. So we use a **BindFilterKey** method to bind the filter key to the **Category.CategoryID** property of the item currently selected in the **Categories** collection.

Finally, we need to specify data binding for the GUI controls in the form (view) **CategoryProductsView.xaml**. Following the MVVM pattern, we do this in XAML (not in code), as shown below:

```
<Grid>
   <Grid.DataContext>
      <local:CategoryProductsViewModel/>
   </Grid.DataContext>
   <Grid.RowDefinitions>
      <RowDefinitionHeight="Auto" />
      <RowDefinition/>
   </Grid.RowDefinitions>
   <ComboBoxName="comboBox1"Margin="5"HorizontalAlignment="Left"Width="186"
      ItemsSource="{BindingCategories}"
      DisplayMemberPath="CategoryName" />
   <sdk:DataGridName="dataGrid1"AutoGenerateColumns="True"Grid.Row="1"
      ItemsSource="{BindingProducts}"/>
</Grid>
```

Defining DataContext like this,

```
   <Grid.DataContext>
      <local:CategoryProductsViewModel/>
   </Grid.DataContext>
```

we made our form create a **CategoryProductsViewModel** object as its data source, and we bound the combo box and the grid to it using {Binding Categories} and {Binding Products}.

Using Entity Framework DataSource in MVVM with other MVVM Framew

ComponentOne Entity Framework DataSource (**EF DataSource**) can be used to build Model-View-View-Model (**MVVM**) applications with any other MVVM frameworks.

ComponentOne Entity Framework DataSource offers several features to make your MVVM development easier:

- **Simplifies MVVM programming**

  Given that **EF DataSource** can be used to simplify MVVM programming, as seen in the Simplifying MVVM topic, it's clearly a tool that can be used for MVVM.

- **Helps create view model classes and alleviate code bloating**

  There are a plethora of tools and frameworks that developers can use to aid them in their work with MVVM, but very few that can help them create view model classes. The majority are designed to help with tasks such as passing commands and messages between the view and view model. Creating view model classes and then synchronizing them with model data is left almost entirely to manual coding. This is the primary cause of code bloating in most MVVM applications and precisely the one that **EF DataSource** is designed to alleviate in a way that is entirely compatible with other frameworks.

- **Allows you to use any framework and live views**

  You can use any framework you like to assist your MVVM application development and simply call on **EF DataSource** to provide live views to help you create view model classes.

To demonstrate these important points, we provide two samples, one using **C1DataGrid** and one using **C1FlexGrid**. The samples can be found in the **ComponentOne Samples\Studio for Silverlight\C1.Silverlight.DataSource\CS** folder:

Silverlight-C1DataGrid\TutorialsSilverlight-C1DataGrid

Silverlight-C1FlexGrid\TutorialsSilverlight-C1FlexGrid

Use the following paths, depending on your operating system:

Windows XP path: C:\Documents and Settings\<username>\My Documents\ComponentOne Samples

Windows 7/Vista path: C:\Users\<*username*>\Documents\.

Essentially all the files in our modified sample are the same as the originals (bar a few cosmetic changes) except one (ViewModels\ViewModel.cs).

In this file, we build the view model class the **EF DataSource** way, using live views. You can see how many re-shaping functions are applied to model data to construct a view model, all done exclusively through LINQ. This made it easy and required little code. The best part is that it synchronizes automatically with model data when data in either of the two layers is changed - no synchronization code was necessary.

The fact that we only changed the way that the view model classes themselves were created (they are still derived from the original base class 'ViewModelBase') and made no other changes to the framework code that Josh Smith had employed in his original sample should serve as an example that **EF DataSource** is entirely compatible with other frameworks. You can continue to use your preferred frameworks when working with MVVM, but now you have an additional tool to make your MVVM development even easier.

ComponentOne Entity Framework DataSource in WPF

ComponentOne **Entity Framework DataSource** (**EF DataSource**) includes a **C1DataSource** control that allows you to specify data sources with rich features in an RAD fashion, mostly by setting properties on a designer surface, with little manual coding. It also includes classes that provide full support of the same features and more: in code, when you need to do it dynamically, at run time, or when finer control over them is required. We'll start with the **C1DataSource** control, and later we will consider the run-time code features as well.

We'll begin our exploration of the **C1DataSource** component by looking at how we can control it via the designer surface. From there, we'll explore how it can be controlled dynamically at run time. Not all features available at run time will be represented here. You can consult the "Programming Guide" of this documentation and the Reference for more runtime features, including client-side transaction support and more.

Simple Binding

We'll begin by creating a new WPF project in Visual Studio.

1. In Visual Studio, choose **File | New | Project.**
2. Select the **WPF Application** template and click **OK**.

Next, add an Entity Data Model based on the Northwind database. This model will provide the data for the entire application:

1. In the **Solution Explorer**, right-click the project name and select **Add | New Item**.
2. In the **Data** category, select the **ADO.NET Data Model** item and then click **Add**.

3. Select **Generate from database** and click **Next**.
4. Create a connection string pointing to the Northwind SQL Server database and click **Next**. In this example, you will use "NORTHWND.mdf", which should be installed in the **Studio for WPF\C1.WPF.DataSource\Data** folder.
5. Select the **Tables** node and click **Finish**.

Now build the project so the new Entity Data Model classes are generated and become available throughout the project.

Next, add a **C1DataSource** component to the application and connect it to the Entity Data Model:

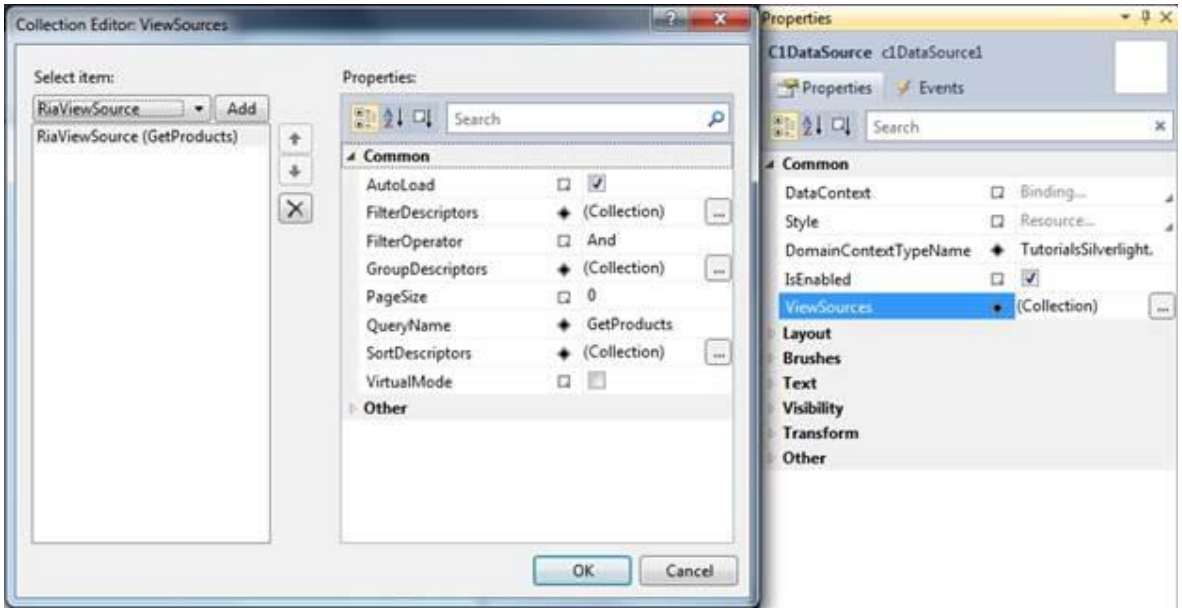
1. Drag a **C1DataSource** component from the Toolbox onto the window. This is a non-visual component, so it can be placed anywhere within the window's inner content.
2. Select the new component and choose **View | Properties Window**.
3. In the Properties window, set the **ObjectContextType** property to the type of object context you want to use. In this case, there should be only one option in the drop-down list, something similar to "AppName.NORTHWINDEntities".

At this point, the **C1DataSource** has created an application-wide object (an **EntityDataCache**) that represents the Northwind database and has application scope. Additional **C1DataSource** objects on other forms will share that same object. As long as they are part of the same application, all **C1DataSource** objects share the same **ObjectContext**.

This unified object context is one of the main advantages **EF DataSource** provides. Without it, you would have to create multiple object contexts throughout the application, and each would have to be synchronized with the others and with the underlying database separately. This would be a non-trivial task, and any errors could compromise the integrity of the data. The unified object context handles that for you transparently. It efficiently caches data and makes it available to all views in a safe, consistent way.

Now that our **C1DataSource** has an **ObjectContext** to work with, we will go on to specify the entity sets it will expose to the application through its **ViewSources** collection. Note that if you are familiar with ADO.NET, you can think of the **C1DataSource** as a **DataSet** and the **ViewSources** collection as **DataView** objects.

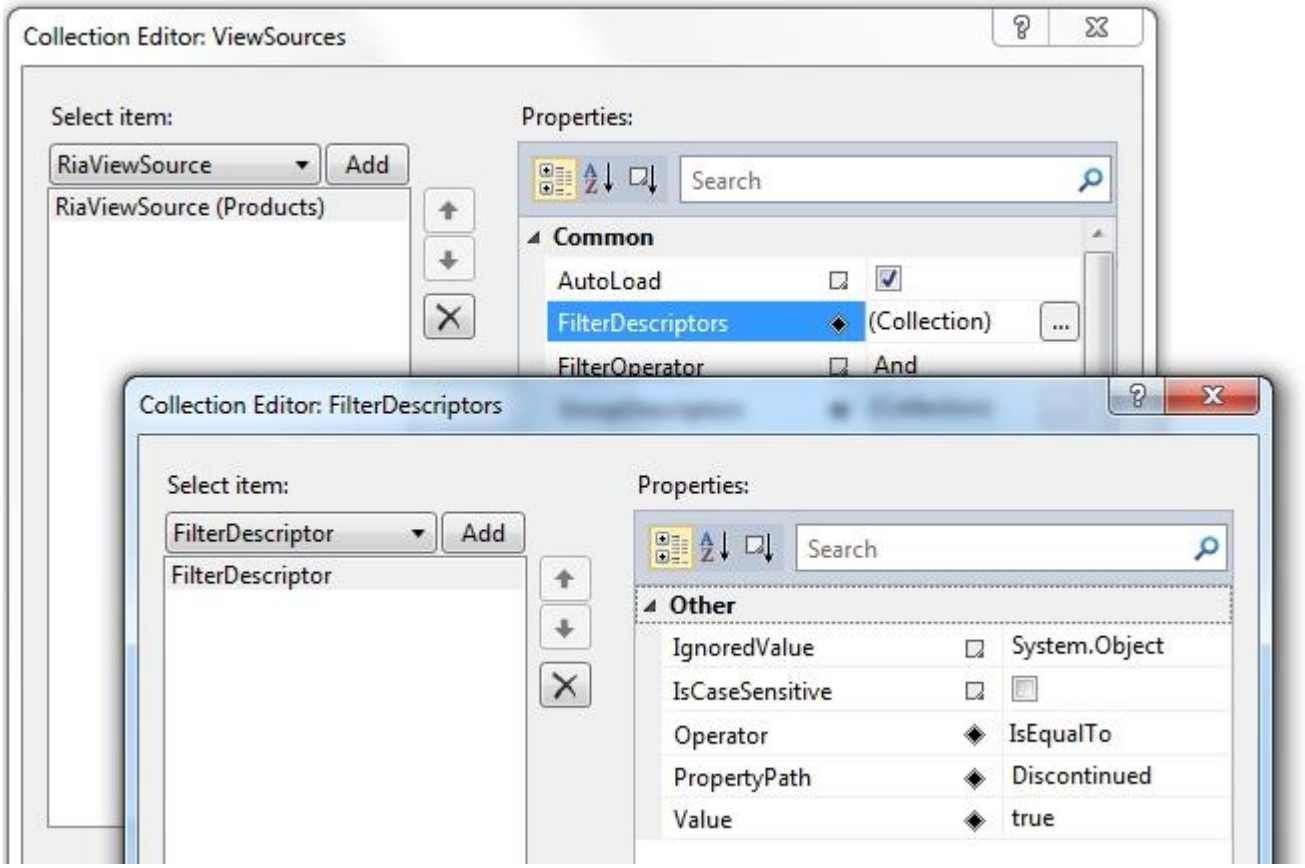
In the properties of the **C1DataSource**, locate the **ViewSourcesCollection** property and open its editor dialog. Click **Add** and then select *Products* from the **EntitySetName** drop-down list.



For this simple example, *Products* is all that is really necessary, but we could continue to create **ViewSources** within this **C1DataSource** in exactly the same way. We might, for example, create a **ViewSource** based on the *Categories* entity set allowing us to have a form that would be used to show the master-detail relationship between *Categories* and their *Products*. Conversely, there is no need to define all of the **ViewSources** that you might need in one single **C1DataSource**. You could have a separate **C1DataSource** for each **ViewSource** that you need. All you need to do is ensure that the **C1DataSource** components that you use utilize the same **ObjectContextType**.

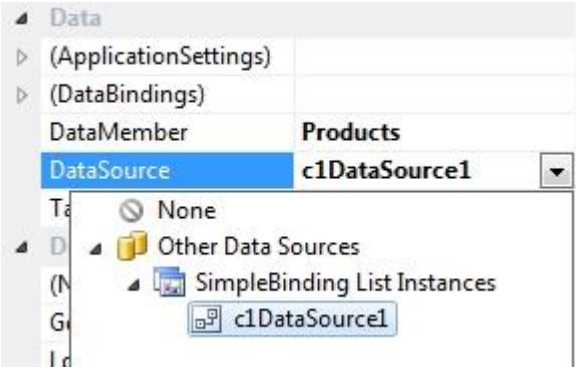
In reality we'll only want to bring a small subsection of the data contained within our database back to the client at any one time. This avoids overloading the client and network and also ensures that we are only presenting data that is relevant to the task our end user is engaged in. The traditional approach to this has been to create code (typically SQL queries) that we would run against the database to achieve our desired results. With **C1DataSource**, we can make use of the designer surface to achieve our goal without the need to write code by specifying server-side filters as property settings.

From the **ViewSourceCollection** editor, open the **FilterDescriptor** collection editor, add a filter descriptor and type the property name and a value for it for server-side filtering. If you need to filter more than one property, you can add additional filter descriptors.



Using exactly the same methodology, you can add **SortDescriptors** and **GroupDescriptors** to provide sorting and grouping on the data you retrieve.

With our **C1DataSource** configured, add a grid control to the form. This can be **DataGridView**, a **C1FlexGrid** or indeed any grid that you are familiar with. Set the grid's **DataSource** property to the name of the **C1DataSource** (if you haven't specifically named the **C1DataSource,** then this will be *c1DataSource1*) and its **DataMember** property to *Products*. The **DataMember** property will, in fact, display a drop-down list of all the **ViewSources** (or Entity Sets) that we defined for the **C1DataSource**.



When you select the **C1DataSource** control as the source of the binding, the binding designer shows you under the **Path** node the list of available entity sets (view sources) to choose from. In this case there is only one, *Products*.

Or you can just type the data binding in XAML:

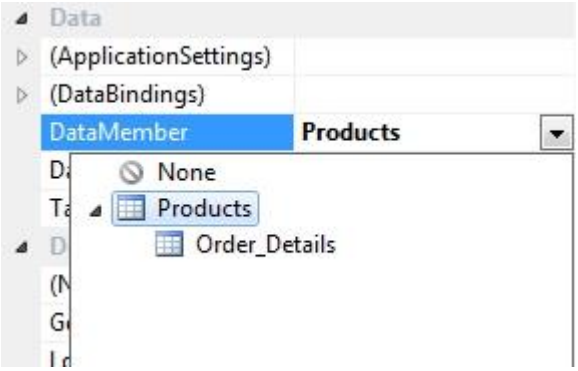
```
ItemsSource="{BindingElementName=c1DataSource1,Path=Products}"
```

At this point, the grid will automatically generate columns for all the fields in the *Product* type, and most grids will allow you to further customize these columns and their layout via their built-in designers.

If you use the standard Microsoft WPF DataGrid, be aware that its **AutoGenerateColumns** property defaults to *False*, so you must either generate columns at design time, or set **AutoGenerateColumns** to *True* in XAML.

Once you are happy with the grid's layout, save, build, and run the application. Notice that the data loads automatically, and you can sort and modify data as you would expect. All this has been achieved by adding just two items to your form (a **C1DatSource** and a data grid) and setting a few properties. You did not have to write a single line of code!

You could continue to add more controls to this form and bind them to specific items in the *Products* collection. To illustrate this point, add a **TextBox** control to the form. From the Properties window, expand the **DataBindings** section and bind its **Text** property to the *ProductName*, as illustrated.



or directly in XAML:

Text="{BindingElementName=c1DataSource1,Path=Products/ProductName}"

Save, build and run the application again, and this time notice how the name of the product currently selected in the grid appears in the **TextBox** that you have just added to the form. If you then edit the product name in either control, the change will be immediately reflected in the other.

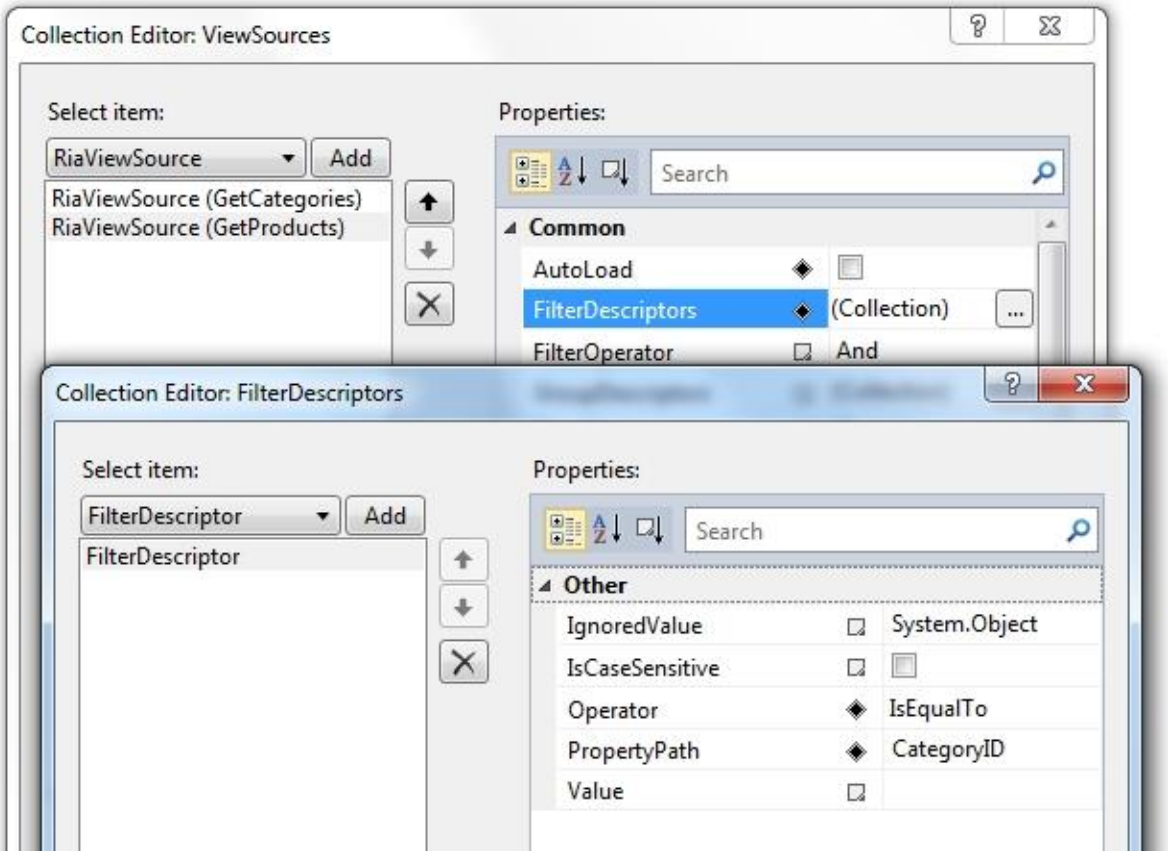
Server-Side Filtering

We already mentioned that it is generally desirable to restrict the data returned from the server to the client and we demonstrated how C1DataSource facilitates this with the use of the **FilterDescriptor Collection Editor**. Now we'll demonstrate how to provide the end user with the means to achieve server-side filtering.

The user will select a *Product Category* from a combo box, for example, although other GUI controls can be used, and that will load a **DataGrid** with new data from the server.

To implement server-side filtering, follow these steps:

1. Add a new form with a **C1DataSource** component to the project used in Simple Binding. You can make this form the project's start up form to save time when you run the project.
2. Establish the **C1DataSource** as before, but this time define two view sources: *Categories* and *Products*. Click the **Add** button twice in the **ViewSourceCollection**. Enter *Categories* next to the EntityViewSourceProperties.EntitySetName property property for the first member (0) and enter *Products* for the second member (1). For *Products*, we'll define a filter as shown in the following picture:



Note that we left **Value** empty. This is because we will be setting it in code in response to a selection change event of the combo box.

3. Bind the grid using **ItemsSource="{Binding ElementName=c1DataSource1, Path=Products}"** as was shown earlier.
4. Set the **AutoGenerateColumns** property of the grid to *True* in XAML. Otherwise the grid will not have any columns at run time.
5.
6. `<DataGrid AutoGenerateColumns="True" …`
7. For the combo box, use the following bindings:
8.
9.
```
ItemsSource="{BindingElementName=c1DataSource1,Path=Categories}"
            DisplayMemberPath="CategoryName"
```

Just as with the grid and other controls, you can use the binding designer in the Properties window where you can choose from lists, or you can enter the binding directly in XAML.

10. Finally, add the following code to the form to handle the combo box's *SelectionChanged* event:

To write code in Visual Basic

Visual Basic

```
Private Sub comboBox1_SelectionChanged(sender As System.Object, e As
System.Windows.Controls.SelectionChangedEventArgs)
    c1DataSource1.ViewSources("Products").FilterDescriptors(0).Value =
        CType(comboBox1.SelectedValue, Category).CategoryID
    c1DataSource1.ViewSources("Products").Load()
 End Sub
```

To write code in C#

C#

```
private voidcomboBox1_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

   c1DataSource1.ViewSources["Products"].FilterDescriptors[0].Value =
      ((Category)comboBox1.SelectedItem).CategoryID;

   c1DataSource1.ViewSources["Products"].Load();

}
```

11. Save, build and run the application. Select a category in the combo box and notice how products related to that category are displayed in the grid. You can still edit the data in the grid exactly as before.

The Power of Client Data Cache

The Server-Side Filtering example shows how the **ComponentOne Entity Framework DataSource** (**EF DataSource**) improves and simplifies working with the Entity Framework in applications, made possible by its client-side data cache, a key feature of **EF DataSource**. It enables several important enhancements and makes writing application code much easier.

Let's start with a performance enhancement that can be seen in the Server-Side Filtering example. When the user switches between categories, the grid loads products associated with them. The first time a category is chosen, there is a slight delay when the relevant data is retrieved. On subsequent occasions, when the same category is chosen, data retrieval is virtually instantaneous. Why? Because the data is being retrieved from the memory EntityDataCache rather than the server.

Note: The EntityDataCache is actually smarter still, determining that a return trip to the server can be avoided, even in more complex cases where the queries may not be identical but can be fulfilled from the results of other queries already contained therein.

This performance enhancement may not be obvious if you are working on a single machine where no network interaction is required, but in real world applications it can make all the difference between a sluggish application that calls the server on every user action and a crisp interface with no delays.

The second important enhancement is in memory management. You might think based on what you've read and observed to date that the EntityDataCache continually accumulates data throughout its existence. If this were the case you would very quickly witness severe performance degradation. In reality the EntityDataCache is keeping track of what is stored within it and as data is no longer required is releasing it performing a self-cleansing operation at the same time. All of this is being done without the need for you to add extra code, and more importantly still it is ensuring that data integrity is being preserved at all times. Data won't be released if required by other data, nor will data that has been altered in some way without those alterations having being saved.

This also relates to performance, because getting rid of unnecessary objects in memory improves performance, and vice versa, keeping large numbers of obsolete objects in memory leads to performance degradation. We mentioned before how **EF DataSource** simplifies context management by eliminating the need to create multiple data contexts thanks to the client cache Now we should explain what the EntityDataCache actually is and how we can further use this information to our advantage.

The EntityDataCache is essentially the context. In terms of the **EF DataSource** namespaces, the cache is the **C1.Data.Entities.EntityClientCache** class, and it is in one-to-one correspondence with **ObjectContext** through its **ObjectContext** property. Both the cache and its underlying **ObjectContext** are created for you automatically if you use the **C1DataSource** control, however, you can create them explicitly and set the **C1DataSource.ClientCache** property in code, if necessary.

To see how simple application code becomes thanks to the client-side cache, let's add the functionality of saving modified data to the project in Server-Side Filtering.

1. Simply add a button to the form and add a handler for the code:

   To write code in Visual Basic

   Visual Basic

   Private Sub button1_Click(sender As System.Object, e As System.Windows.RoutedEventArgs)

```
    c1DataSource1.ClientCache.SaveChanges()
End Sub
```

To write code in C#

C#

```
private voidbutton1_Click(object sender, RoutedEventArgs e)

{

    c1DataSource1.ClientCache.SaveChanges();

}
```

2. Save, build and run the application.

- Select a category and then make some changes to the product information in the grid.
- Select another category (and possibly a third) and again makes changes to the product details in the grid.
- Click the button you added, close the application, reopen it and select the same categories as before.
- Observe how the changes you made have been saved. **EF DataSource** has provided, via the EntityDataCache, a way to alter the product details of several categories' products without the need to save those changes each time different category is selected. To achieve this effect without writing a lot of code, you'd need to keep the entities (the product details) from different categories in the same context. This would waste memory without releasing it, creating a memory leak. **EF DataSource** simplifies all of this for you while optimizing memory usage and performance.

Client-side caching also makes possible other important features of **EF DataSource**, such as client-side queries and, especially, live views. Live Views is a feature that allows you to replace much of the complex application coding with simple data binding, which we'll learn more about later.

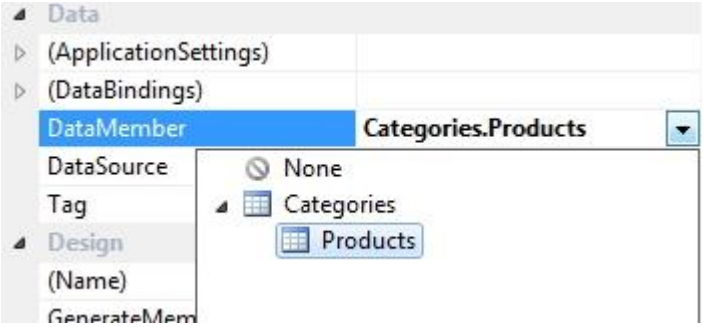
Master-Detail Binding

As we've already seen in the Server-Side Filtering example, C1DataSource supports master-detail binding. With very large datasets, server-side filtering is by far the best solution, but with smaller

datasets, client-side filtering can be just as efficient. This next scenario demonstrates client-side master-detail binding using a grid, instead of a combo box like in our previous example, to select our categories.

To implement master-detail binding, follow these steps:

1. Using the project we created to demonstrate Server-Side Filtering, add a new form with a **C1DataSource** component using the same ObjectContextType as before and create a ViewSource based on *Categories*. Note that you can make this the startup form to save time when you run the project.
2. Next, add the master grid to the form and bind it to *Categories*:
3.
4. `ItemsSource="{BindingElementName=c1DataSource1,Path=Categories}"`
5. Now add a second grid to the form below the one you've just configured. Its **DataSource** will again be the **C1DataSource**, but set its **DataMember** property to the *Products* node which you'll find underneath *Categories* as shown in the following picture:



Or you can type it directly in XAML:

```
ItemsSource="{BindingElementName=c1DataSource1,Path=Categories/Products}"
```

6. Save, build and run the application.

This form of the **Path** parameter in binding, using "/"", Path=Categories/Products, causes the grid to automatically show the products that belong to the category that is currently selected. Running this on a single machine, as you probably are, you won't notice any significant time lapse as you select new categories in the master grid, and their respective products are displayed in the details grid. In the background, the **C1DataSource** is making use of an Entity Framework feature, implicit lazy loading, meaning that products are only being summoned for new categories as they are selected. For many scenarios, this is perfectly acceptable, but we began this section by specifically referring to master-detail relationships in small datasets. We might just as well fetch all of the products for all of the

categories when the form loads and then display will be instantaneous whether on a single machine or across a network. To achieve this behavior, open the **ViewSourceCollection** editor and type *Products* in the **Include** property of the *Categories* view source.

Large Datasets: Paging

To show large amounts of data without bringing it all to the client at once, applications have traditionally used paging. Paging is not an ideal solution; it complicates the interface and makes it less convenient for the user, but it is preferred in some applications. For these cases, C1DataSource supports paging as expected, but it does so without the traditional limitations on data modification. The user can make changes in multiple pages in one session without being forced to send the changes from one page to the database before moving to the next page. That's a substantial enhancement compared to other paging implementations, such as the one in DomainDataSource in Microsoft RIA Services.

Note: EF DataSource does offer a solution to the drawbacks of paging; we'll cover virtual mode later in this documentation.

To implement paging, follow these steps:

1. Using the project we created to demonstrate Master-Detail Binding, add a new form with a **C1DataSource** component using the same **ObjectContextType** as before. Note that you can make this the startup form to save time when you run the project.
2. Create a **ViewSource** in the **ViewSourceCollection** editor, entering *Orders* as the EntitySetName.
3. To enable paging, set the **PageSize** property to 10 for now, but you can choose any reasonable value for this property. It's simply determining the number of data rows that will be displayed on a page.
4. Next, add a grid to the form and bind it to *Orders* the same way we did before:
5.
6. `ItemsSource="{Binding ElementName=c1DataSource1, Path=Orders}"`
7. Set the **AutoGenerateColumns** property of the grid to *True* in XAML.
8. To allow the user to move between pages, and to show the current page and page count, we need to add a few more controls. Add two buttons and a label using, for example, the following XAML:
9.

```
10. <StackPanel Orientation="Horizontal" Grid.Row="1" Margin="2">
11.    <Button Padding="10,0,10,0" Margin="1" Content="&lt;"
    Click="MoveToPrevPage"/>
12. 
13.    <TextBlock VerticalAlignment="Center" Text="Page: "/>
14. 
15.    <TextBlock VerticalAlignment="Center" x:Name="pageInfo"/>
16. 
17.    <Button Padding="10,0,10,0" Margin="1"  Content="&gt;"
    Click="MoveToNextPage"/>
18. 
19. </StackPanel>
```

20. Add the following code containing a **RefreshPageInfo** handler for the **PropertyChanged** event used to show current page number and page count, and handlers for the button **Click** events used to move to the next and previous pages:

To write code in Visual Basic

Visual Basic

```
Imports C1.Data.DataSource

 Public Class Paging
    Private _view As ClientCollectionView

    Public Sub New()

       InitializeComponent()

       _view = C1DataSource1("Orders")

       RefreshPageInfo()

       AddHandler _view.PropertyChanged, AddressOf RefreshPageInfo

    End Sub

    Private Sub RefreshPageInfo()
       pageInfo.Text = String.Format("{0} / {1}", _view.PageIndex + 1, _view.PageCount)
    End Sub

    Private Sub btnPrevPage_Click(sender As System.Object, e As System.EventArgs)
Handles btnPrevPage.Click
       _view.MoveToPreviousPage()
    End Sub

    Private Sub btnNextPage_Click(sender As System.Object, e As System.EventArgs)
Handles btnNextPage.Click
       _view.MoveToNextPage()
    End Sub
 End Class
```

To write code in C#

C#

```
using C1.Data.DataSource;


namespaceTutorialsWPF
{
    public partial class Paging : Window
    {
        ClientCollectionView _view;


        public Paging()
        {
            InitializeComponent();


            _view = c1DataSource1["Orders"];


            RefreshPageInfo();
            _view.PropertyChanged += delegate { RefreshPageInfo(); };
        }


        private void RefreshPageInfo()
        {
            labelPage.Text = string.Format("{0} / {1}",
         _view.PageIndex + 1, _view.PageCount);
        }


        private void MoveToPrevPage(object sender, RoutedEventArgs e)
```

```
        {
            _view.MoveToPreviousPage();
        }

        private void MoveToNextPage(object sender, RoutedEventArgs e)
        {
            _view.MoveToNextPage();
        }
    }
}
```

21. Save, build and run the application. Page through the Orders. While you are moving between pages, try changing some data in the grid. Try changing data in one page, and then move to another page and try changing data there. Notice that **C1DataSource** allows you to do this without forcing you to save data to the database before leaving the page. This is an important enhancement compared with other paging implementations, including the one supported by DomainDataSource in Microsoft RIA Services (which is for Silverlight only, whereas **EF DataSource** supports this, as other features, for all three platforms: WinForms, WPF, Silverlight).

    Try also to delete some orders. This is also allowed without restrictions, and moreover, the current page will automatically complete itself to keep the number of rows in the page unchanged.

    And if you add a **Save Changes** button, using the following code as we did previously, then you will be able to save changes made in multiple pages by pressing that button when you are done.

    To write code in Visual Basic

    Visual Basic

```
Private Sub btnSaveChanges_Click(sender As System.Object, e As
System.Windows.RoutedEventArgs)
    c1DataSource1.ClientCache.SaveChanges()
```

```
End Sub
```

To write code in C#

C#

```
private void btnSaveChanges_Click(object sender, RoutedEventArgs e)
{
    c1DataSource1.ClientCache.SaveChanges();
}
```

All this functionality is what you would expect from a paging implementation that supports unrestricted data modification. Unfortunately, it is not easy to implement. For example, think of all possible cases where changing data on one page interferes with what should be shown on other pages, and so on. That is why paging usually imposes severe restrictions on data modifications, if they are at all allowed. For example, Microsoft DomainDataSource requires you to save all changes before changing pages. Fortunately, **EF DataSource** supports paging without restricting data modifications in any way.

As with many other features, unlimited modifiable paging is made possible by the client-side cache, see The Power of Client Data Cache. That also means that paging implementation in **EF DataSource** is optimized both for performance and for memory consumption. The cache provides that optimization. It keeps recently visited pages in memory, so re-visiting them is usually instantaneous. And it manages memory resources, releasing old pages when necessary, to prevent memory leaks.

Large Datasets: Virtual Mode

As mentioned in the Large Datasets: Paging topic, **ComponentOne Entity Framework DataSource** (**EF DataSource**) has an even better solution than paging for dealing with large data and large numbers of rows.

What if using large datasets with thousands, or even millions, of rows was no longer a problem? What if you could use the same controls to display massive datasets as you would for smaller datasets, with no paging, no code changes, and all by setting one Boolean property? With **EF DataSource** you can, thanks to the magical **VirtualMode** property.

To implement virtual mode, follow these steps:

1. Using the project we created to demonstrate Large Datasets: Paging, add a new form with a **C1DataSource** component using the same **ObjectContextType** as before. Note that you can make this the startup form to save time when you run the project.
2. Create a **ViewSource** in the **ViewSourceCollection** editor. Use the largest table in our sample database: *Order_Details*.
3. Set the **VirtualMode** property to *Managed*.Another possible value is *Unmanaged*, but that is an advanced option that should be used with caution and only when necessary. The *Managed* option means that getting data from the server is managed by a grid control. With the Managed option, EF DataSource supports all main Microsoft and ComponentOne grid controls: C1FlexGrid, C1DataGrid, and Microsoft DataGrid for WPF . It is optimized for performance with those specific grid controls. The *Unmanaged* option means that virtual mode is not driven by a specific control, will work with any bound controls but is subject to some limitations described here.
4. Add a grid to the designer and bind it to *Order_Details* the same way we did it before:
5.
6. `ItemsSource="{BindingElementName=c1DataSource1,Path=Order_Details}"`
7. Set the **AutoGenerateColumns** property of the grid to *True* in XAML.
8. Now, since we selected the *Managed* option, we need to specify which grid control is driving the data. **EF DataSource** defines an attached property **C1DataSource.ControlHandler** that is an object having properties affecting the control's behavior when it is bound to a C1DataSource. There is a boolean **VirtualMode** property in **C1DataSource.ControlHandler** that marks the control as the main, "driving" control in *Managed* virtual mode. Add the following inside the DataGrid markup in XAML (the added markup is shaded):
9.
10.
```
<DataGrid AutoGenerateColumns="True"  Name="dataGrid1"
                 ItemsSource="{Binding ElementName=c1DataSource1,
Path=Order_Details}">
                    <c1:C1DataSource.ControHandler>
                           <c1:ControlHandler VirtualMode="True"/>
                     </c1:C1DataSource.ControHandler>
                </DataGrid>
```
11. Save, build and run the application. You'll see nothing earth-shattering, simply a grid you can navigate, scroll and modify row data as you see fit. It looks and behaves like any conventional data grid, which is precisely the point. Now you can use large datasets free of the drawbacks associated with paging, and all done without the need to write any code. And as an added benefit, you can use any GUI control you like so long as it has a **DataSource** property. This example has used a relatively modest-sized dataset, but **C1DataSource** running in virtual mode would be just as responsive with a much larger dataset; it does not depend on the number of rows. To further prove the point, look at the **OrdersDemo** sample installed with this product. The sample uses a larger database, with roughly 65,000 additional rows, but the responsiveness is the same as our example here. Again, it does not depend on the number of rows in the dataset.

How does this magic work? It works much like paging, only hiding its inner workings from the GUI controls, with paging occurring under the hood. The GUI controls see the data as if it is fetched to the client and ready for them. When GUI controls request data, **C1DataSource**, or **ClientViewSource** if it is used in code without a **C1DataSource** control, first checks whether it can serve the data from memory, from the same client-side cache it uses for all features. If it can't find it

in memory, it transparently fetches the required data from the server. As with other features using the client-side cache, **C1DataSource** does not store the fetched data indefinitely, which would be a memory leak. It knows which parts of the data are needed for serving the GUI controls and which parts should be kept because they are modified or related to modified parts, and so on. It releases old, unnecessary parts of data as needed. No code is required and any GUI controls can be used.

Automatic Lookup Columns in Grids

A common scenario in data binding is for data classes to contain references to other data classes. For example, a **Product** object may contain references to **Category** and **Supplier** objects.

In ADO.NET, the references usually appear as foreign keys that map in other tables (e.g., **Product.CategoryID** and **Product.SupplierID**).

In the Entity Framework, you still get the key columns, but you also get the actual objects. So you have **Product.CategoryID** (usually an integer) and **Product.Category** (an actual Category object).

Displaying foreign keys in a grid is not very useful, because it is unlikely that users will remember that category 12 is "Dairy Products" or that supplier 15 is "ACME Imports". Allowing users to edit these keys would be even worse. A common way to work around this problem is to remove the related entity columns from any bound grids and, optionally, to replace them with custom columns that use combo boxes for editing the values, so called lookups. The combo boxes have to be bound to the related tables and have their properties set so they display relevant values (e.g., **Category.Name** or **Supplier.CompanyName**) and so they are synchronized with the values being displayed on the grid. This is not terribly hard to do, but it is a tedious and error-prone task that makes projects harder to create and maintain.

C1DataSource can do this tedious work for the developer; it can automatically change related entity columns so that they show combo box lookups. It can do this for several types of data grids, those that it supports. Currently supported WPF grids are: C1FlexGrid, C1DataGrid, and Microsoft DataGrid for WPF. Here we will show how to do this for **C1FlexGrid**.

C1DataSource provides an *attached* property called **ControlHandler**. If you place a **C1FlexGrid** control on a window designer that contains a **C1DataSource**, you can set an additional **C1DataSource.ControlHandler** property on the grid. A ControlHandler is an object containing (at present) a single boolean property **AutoLookup**. This property set to *True* causes the **C1DataSource** to configure grid columns that contain references to other entities so that they show lookup combos.

To see how it works, follow these steps:

1. Using the project used in Simple Binding, add a new form with a **C1DataSource** control using the same **ObjectContextType** as before.

2. Create a **ViewSource** in the **ViewSourceCollection** editor, entering *Products* as the **EntitySetName**.
3. Add a **C1FlexGrid** to the designer and bind it to *Products* as we did it before:
4.
5. `ItemsSource="{BindingElementName=c1DataSource1,Path=Products}"`

6. Save, build and run the application. It will look like this:

Product ID ▲	Product Name	Category
5	Chef Anton's Gumbo Mix	Category : 2
9	Mishi Kobe Niku	Category : 6
17	Alice Mutton	Category : 6
24	Guaraná Fantástica	Category : 1
28	Rössle Sauerkraut	Category : 7
29	Thüringer Rostbratwurst	Category : 6
42	Singaporean Hokkien Fried Me	Category : 5
53	Perth Pasties	Category : 6

As you can see, the **Category** and **Supplier** columns are not useful at all. You could remove them or customize the grid by writing some code to create new columns, but there's an easier way.

7. Add the following inside the C1FlexGrid markup in XAML (the added markup is shaded):
8.
9.
```
<c1:C1FlexGrid
AutoGenerateColumns="True"Name="dataGrid1"ItemsSource="{BindingElementNam
e=c1DataSource1,Path=Products}">
                    <c1:C1DataSource.ControHandler>
                          <c1:ControlHandler AutoLookup="True"/>
                    </c1:C1DataSource.ControHandler>
        </c1:C1FlexGrid>
```
10. Run the project again and look at the **Category** and **Supplier** columns. Notice that now the grid shows the category name and supplier's company name instead of the generic strings. Also notice that you can edit the product's category and the product's supplier by picking from a drop-down list, complete with auto search functionality.

11. Finally, click the column headers to sort the grid by **Category** or **Supplier** and notice that the sorting is performed based on the value displayed on the grid. This is what anyone would expect, but surprisingly it is not easy to achieve using regular data binding.

The string value (name) shown by the combo box is determined following these rules:

1. If the entity class overrides the **ToString** method, then the string representation of the entity is obtained using the overridden **ToString** method. This should return a string that uniquely represents the entity. For example, it could be the content of a **CompanyName** column, or a combination of **FirstName**, **LastName**, and **EmployeeID**. This is the preferred method because it is simple, flexible, and easy to implement using partial classes (so your implementation will not be affected if the entity model is regenerated).
2. If the entity class does not override the **ToString** method, but one of its properties has the **DefaultProperty** attribute, then that property is used as the string representation of the entity.
3. If the entity class does not override the **ToString** method and has no properties marked with the **DefaultProperty** attribute, then the first column that contains the string "Name" or "Description" in its name will be used as a string representation for the entities.
4. If none of the above applies, then no lookup is created for the given entity type.

Customizing View

In many situations, you may want to use custom views that do not correspond directly to the tables and views provided by the database. LINQ is the perfect tool for these situations allowing you to transform raw data using your language of choice using query statements that are flexible, concise, and efficient. **ComponentOne Entity Framework DataSource (EF DataSource)** makes LINQ even more powerful by making LINQ query statements *live*. That's why its LINQ implementation is called LiveLinq. We will show the power of LiveLinq in Live Views. For now, suffice it to say that you can transform your view to shape it whatever way you want using LINQ operators without losing full updatability and bindability.

Let's try, for example, one of the LINQ operators: **Select** (also called projection), to customize the fields (properties) of our view.

To customize a view, follow these steps:

1. Using the project we created to demonstrate Paging, add a new form with a **C1DataSource** component using the same **ObjectContextType** as before. Note that you can make this the startup form to save time when you run the project.
2. Create a **ViewSource** in the **ViewSourceCollection** editor. Use the *Products* table in our sample database.
3. Add a grid to the window designer and, as before, set its **AutoGenerateColumns** property to *True*. But this time we won't bind the grid to **C1DataSource** in XAML. Instead, we will bind it in code, because we will create a custom view in code.
4. Create a custom live view and bind the grid to that view with the following code:

   To write code in Visual Basic

   Visual Basic

```
dataGrid1.ItemsSource = _
   (From p In c1DataSource1("Products").AsLive(Of Product)()
    Select New With
    {
        p.ProductID,
        p.ProductName,
        p.CategoryID,
        p.Category.CategoryName,
        p.SupplierID,
        .Supplier = p.Supplier.CompanyName,
        p.UnitPrice,
        p.QuantityPerUnit,
        p.UnitsInStock,
        p.UnitsOnOrder
     }).AsDynamic()
```

   To write code in C#

   C#

```
dataGrid1.ItemsSource =

   (from p in c1DataSource1["Products"].AsLive<Product>()
```

```
select new
{
    p.ProductID,
    p.ProductName,
    p.CategoryID,
    CategoryName = p.Category.CategoryName,
    p.SupplierID,
    Supplier = p.Supplier.CompanyName,
    p.UnitPrice,
    p.QuantityPerUnit,
    p.UnitsInStock,
    p.UnitsOnOrder
}).AsDynamic();
```

Here c1DataSource1["Products"] is a ClientCollectionView object. It is the view that is created by the view source that we set up in the designer (if you need to access the view source itself in code, it is also available, as c1DataSource.ViewSources["Products"]).
The AsLive() method call is needed to specify the item type of the view (*Product*) so LiveLinq operators can be applied to it. The result, c1DataSource1["Products"].AsLive<Product>(), is a View<Product>. The C1.LiveLinq.LiveViews.View is the main class of LiveLinq, used for client-side live views. LINQ operators applied to live views preserve their updatability and bindability. They are the same usual LINQ operators, but the fact that they are applied to a live view gives them these additional features that are critically important for data binding applications.

Note: AsDynamic() must be applied to this view because its result selector (the select new... code) uses an anonymous class. This is a minor LiveLinq limitation, only for anonymous classes. If you forget to add AsDynamic() to such view, you will be reminded by an exception.

5. Save, build and run the application. The grid now shows the columns we defined in the 'select' clause of our LiveLinq view. Note also that all columns are modifiable. It may not seem like a big deal; however, it is an important feature that would be difficult to implement on your own for this customized view, especially when adding and deleting rows, which, as you can see here, is also supported. To try deleting a row, just select a row and press the **Delete** key. To try adding a row, you'll need to add a button to the window executing the following code:

To write code in Visual Basic

Visual Basic

```
Dim newItem = CType(dataGrid1.ItemsSource,
System.ComponentModel.IEditableCollectionView).AddNew()
    dataGrid1.ScrollIntoView(dataGrid1.Rows.Count - 1, 0)
```

To write code in C#

C#

```
object newItem =

  ((System.ComponentModel.IEditableCollectionView)dataGrid1.ItemsSource).AddNew();

dataGrid1.ScrollIntoView(newItem);
```

This is needed only because Microsoft WPF DataGrid does not provide a built-in interface for adding new rows.

Bindability is achieved because the views are always 'live'; they aren't simple snapshots of static data. To prove the point, construct an exact replica of the form that we've just built. At this stage, you might find it easier to add a menu to your application to make accessing the forms you've created easier. Save, build and run the application. This time, open two instances of the form you just created and change some data in the grid on one of the forms. Notice how the corresponding data in the grid on the other form is changed automatically. Remember, you have not had to write a single line of code for the grids in these forms to synchronize the changes that you've just made.

You will see more of what LiveLinq can do in the Live Views topic.

Working with Data Sources in Code

Up to this point, we have been setting up data sources directly on the designer surface with very little code. **ComponentOne Entity Framework DataSource (EF DataSource)** has made it very easy, but sometimes you want or need to do everything in code. **EF DataSource** makes this possible as well. Everything we did previously can be done at run time in code.

An obvious way to go about this would be to use the **ClientViewSource** object that we have, in effect, been setting up in the designer as elements of the **ViewSourceCollection** of a **C1DataSource**, given that it can be created on its own without a **C1DataSource**. We could, however, take a step further back and use a lower level class ClientView<T>. This would provide full control over loading data from the server and, since it is derived from C1.LiveLinq.LiveViews.View<T>, we can apply any LiveLinq operators to it. The ability to bind this to any GUI control whose datasource can be set to a View<T> also means that we'll end up with a fully editable view of our data.

Server-side filtering is, perhaps, the most common operation, because no one usually wants entire database tables brought to the client unrestricted. Earlier we saw how **EF DataSource** made it simple to perform without code, but now we'll try it in run-time code.

To begin using **EF DataSource** in run-time code without a **C1DataSource**, add a few lines to our project's main class to create a global client-side data cache. When we used **C1DataSource**, it was created for us behind the scenes. Now we create it explicitly using the following code:

To write code in Visual Basic

Visual Basic

```
Imports C1.Data.Entities

 Class Application

    Public Shared ClientCache As EntityClientCache

    Private Sub Application_Startup(sender As Object, e As System.Windows.StartupEventArgs)
Handles Me.Startup

        ClientCache = EntityClientCache.GetDefault(GetType(NORTHWNDEntities))

    End Sub
 End Class
```

To write code in C#

C#

```
using C1.Data.Entities;


public partial classApp : Application
{
    public static EntityClientCache ClientCache;
    public static NORTHWNDEntities ObjectContext;


    protected override void OnStartup(StartupEventArgs e)
    {
        base.OnStartup(e);


        ObjectContext = new NORTHWNDEntities();
        ClientCache = new EntityClientCache(ObjectContext);
    }
}
```

This code creates a single application-wide (static) **ObjectContext** and associates it with **EntityClientCache**. As noted previously in The Power of Client Data Cache topic, the ability to have a single context (and cache) for the entire application is a great simplification made possible by **EF DataSource**.

To perform server-side filtering in run-time code, follow these steps:

1. Create a new window, add a grid (**dataGrid1** and set its **AutoGenerateColumns** property to *True*), a combo box (**comboBox1**), and a button (**btnSaveChanges**) to the form, and add the following code to the window class:

   To write code in Visual Basic

Visual Basic

```
Imports C1.Data.Entities
 Imports C1.Data

 Public Class DataSourcesInCode

    Private _scope As EntityClientScope


    Public Sub New()

       ' This call is required by the designer.
       InitializeComponent()

       ' Add any initialization after the InitializeComponent() call.

       _scope = Application.ClientCache.CreateScope()

       Dim viewCategories As ClientView(Of Category) = _scope.GetItems(Of Category)()

       comboBox1.DisplayMemberPath = "CategoryName"
       comboBox1.ItemsSource = viewCategories

       BindGrid(viewCategories.First().CategoryID)
    End Sub

    Private Sub BindGrid(categoryID As Integer)
       dataGrid1.ItemsSource =
             (From p In _scope.GetItems(Of Product)().AsFiltered(
                 Function(p As Product) p.CategoryID.Value = categoryID)
              Select New With
              {
                 p.ProductID,
                 p.ProductName,
                 p.CategoryID,
                 p.Category.CategoryName,
                 p.SupplierID,
                 .Supplier = p.Supplier.CompanyName,
                 p.UnitPrice,
                 p.QuantityPerUnit,
                 p.UnitsInStock,
                 p.UnitsOnOrder
              }).AsDynamic()
    End Sub
```

```
    Private Sub comboBox1_SelectionChanged(sender As System.Object, e As
System.Windows.Controls.SelectionChangedEventArgs) Handles
comboBox1.SelectionChanged
        If comboBox1.SelectedValue IsNot Nothing Then
            BindGrid(CType(comboBox1.SelectedValue, Category).CategoryID)
        End If
    End Sub

    Private Sub btnSaveChanges_Click(sender As System.Object, e As
System.Windows.RoutedEventArgs) Handles btnSaveChanges.Click
        Application.ClientCache.SaveChanges()
    End Sub

 End Class
```

To write code in C#

C#

```
using C1.Data.Entities;

using C1.Data;


public partial classDataSourcesInCode : Window

{

    private EntityClientScope_scope;


    publicDataSourcesInCode()

    {

        InitializeComponent();


        _scope = App.ClientCache.CreateScope();
```

```
        ClientView<Category> viewCategories = _scope.GetItems<Category>();

        comboBox1.DisplayMemberPath = "CategoryName";
        comboBox1.ItemsSource = viewCategories;

        BindGrid(viewCategories.First().CategoryID);
    }

    private void comboBox1_SelectionChanged(object sender, SelectionChangedEventArgs
e)
    {
        if (comboBox1.SelectedValue != null)
            BindGrid(((Category)comboBox1.SelectedValue).CategoryID);
    }

    private void BindGrid(int categoryID)
    {
        dataGrid1.ItemsSource =
            (from p in _scope.GetItems<Product>().AsFiltered(
                 p => p.CategoryID == categoryID)
             select new
             {
                 p.ProductID,
                 p.ProductName,
                 p.CategoryID,
```

```
                CategoryName = p.Category.CategoryName,
                p.SupplierID,
                Supplier = p.Supplier.CompanyName,
                p.UnitPrice,
                p.QuantityPerUnit,
                p.UnitsInStock,
                p.UnitsOnOrder
            }).AsDynamic();
    }

    private void btnSaveChanges_Click(object sender, RoutedEventArgs e)
    {
        App.ClientCache.SaveChanges();
    }
}
```

2. Save, build and run your application. You should see similar results to those you saw in the Server-Side Filtering example, the only difference being that this time we've implemented it all in code.

   The private field **_scope** is the form's gateway to the global data cache. It is a pattern we recommend you follow in all the forms where you do not employ a **C1DataSource** component directly, as that does this for you automatically. It ensures that the entities the form needs stay in the cache while the form is alive, and that they are automatically released when all forms (scopes) holding them are released.

   Creating a view showing all categories for the combo box is simple:

   To write code in Visual Basic

Visual Basic

```
Dim viewCategories As ClientView(Of Category) = _scope.GetItems(Of Category)()
```

To write code in C#

C#

```
ClientView<Category> viewCategories = _scope.GetItems<Category>();
```

To create the view to which the grid is bound that only provides those products associated with the chosen category in the combo box required one additional operator; AsFiltered(<predicate>).

To write code in Visual Basic

Visual Basic

```
From p In _scope.GetItems(Of Product)().AsFiltered(Function(p As Product)
p.CategoryID.Value = categoryID)
```

To write code in C#

C# **Copy Code**

```
from p in _scope.GetItems<Product>().AsFiltered(p => p.CategoryID == categoryID)
```

Note that when this query is executed, the result does not necessarily require a round trip to the server to retrieve the products requested. The cache is examined first to see if it already contains the requested data, either because the required data has already been requested once before within this form or from another form in the application. Or, possibly a completely separate query run elsewhere in the application had requested that all products be returned, so the cache would already have all the product data. Again, this is a fundamental strength of **EF DataSource**. By providing your application with a global cache of data, its performance is continually improved throughout its lifetime.

Here we chose to create a new view, and bind the grid to it, every time the user selects a new category in the combo box (see the combo box's **SelectedValueChanged** event). However, we could have avoided the need to create new views all the time and instead created one single view using a special **BindFilterKey**, which we'll learn more about in the Simplifying MVVM topic.

So, in summary, we replicated in code what we did on the design surface with **C1DataSource** in Server-Side Filtering. We have even thrown in a little extra; we customized the fields shown in the grid columns as we did in Customizing View by adding a **Select** to our LiveLinq statement:

To write code in Visual Basic

Visual Basic

```
Select New With
 {
    p.ProductID,
    p.ProductName,
    p.CategoryID,
    p.Category.CategoryName,
    p.SupplierID,
    .Supplier = p.Supplier.CompanyName,
    p.UnitPrice,
    p.QuantityPerUnit,
    p.UnitsInStock,
    p.UnitsOnOrder
 ;
```

To write code in C#

C#

```
select new

{

   p.ProductID,

   p.ProductName,

   p.CategoryID,
```

```
    CategoryName = p.Category.CategoryName,
    p.SupplierID,
    Supplier = p.Supplier.CompanyName,
    p.UnitPrice,
    p.QuantityPerUnit,
    p.UnitsInStock,
    p.UnitsOnOrder
};
```

Had we just wanted the raw product data returned from the table without any special formatting, we could have simply said;

```
select p;
```

Live Views

Live views is a powerful feature, so let's take a little more time to see what live views can do. Live views are designed to make data binding more powerful, so powerful, in fact, that you can develop virtually entire applications with just LINQ (in its LiveLinq form) and data binding.

Live views, called **LiveLinq**, a part of **ComponentOne Entity Framework DataSource** (**EF DataSource**), is a client-side, in-memory feature applicable not only to Entity Framework and RIA Services data, but to any observable collections in memory, including XML (LINQ to XML object model) and ADO.NET DataSets.

So, for example, you can use live views over Entity Framework data and some other data (for example, XML retrieved from some web service) to integrate that data and to provide easy full-featured data binding to the integrated data. This is a powerful tool for building applications that get data from various sources. For now we're going to concentrate on how we can use it with the Entity Framework, but if you'd like to explore **LiveLinq** in greater depth, see the ComponentOne LiveLinq documentation.

In fact, we already saw an example of such customization in Customizing View. But there we only changed the properties (fields) of the view and only applied one LINQ operator, **Select**. Let's apply

some more LINQ operators to transform the view. What we do here will be similar to what we did in Customizing View, but, instead of using **C1DataSource**, we'll be doing everything in code.

To use live views, follow these steps:

1. Add a new form using the project created to demonstrate Working with Data Sources in Code, and add a data grid, **dataGridView1**, to the form.
2. In the form's Load event, add the following code. Here we are following the pattern recommended in Working with Data Sources in Code.

   To write code in Visual Basic

   Visual Basic

   ```
   Private _scope As EntityClientScope = Application.ClientCache.CreateScope()
   ```

   To write code in C#

   C#

   ```
   private EntityClientScope_scope = Application.ClientCache.CreateScope();
   ```

3. Getting *Products* data from the scope, we create a live view and bind the grid to that view in the form's constructor:

   To write code in Visual Basic

   Visual Basic

   ```
   _viewProducts =
      (From p In _scope.GetItems(Of Product)()
        Where Not p.Discontinued And p.UnitPrice >= 30
        Order By p.UnitPrice
        Select  New With
        {
           p.ProductID,
           p.ProductName,
           p.CategoryID,
           p.Category.CategoryName,
           p.SupplierID,
   ```

```
        .Supplier = p.Supplier.CompanyName,
        p.UnitPrice,
        p.QuantityPerUnit,
        p.UnitsInStock,
        p.UnitsOnOrder
    }).AsDynamic()
dataGrid1.ItemsSource = _viewProducts
```

To write code in C#

C#

```
_viewProducts =

    (from p in _scope.GetItems<Product>()

     where !p.Discontinued && p.UnitPrice >= 30

     orderby p.UnitPrice

     select new

     {

        ProductID = p.ProductID,

        ProductName = p.ProductName,

        CategoryID = p.CategoryID,

        CategoryName = p.Category.CategoryName,

        SupplierID = p.SupplierID,

        Supplier = p.Supplier.CompanyName,

        UnitPrice = p.UnitPrice,

        QuantityPerUnit = p.QuantityPerUnit,

        UnitsInStock = p.UnitsInStock,

        UnitsOnOrder = p.UnitsOnOrder

     }).AsDynamic();
```

```
dataGrid1.ItemsSource = _viewProducts;
```

In this example, we applied several **LiveLinq** operators: *Where*, *OrderBy*, and *Select*. We defined our view as containing products that aren't discontinued and have a unit price of at least 30, and we sorted our view by unit price.

We chose to store the view in a private field _viewProducts here:

To write code in Visual Basic

Visual Basic

```
Private _viewProducts As View(Of Object)
```

To write code in C#

C#

```
private View<dynamic> _viewProducts;
```

That is only because we will need it later. If we did not, we could use a local variable for the view.

Syntactically, the query that we wrote for **_viewProducts** is just standard LINQ. It could be done without **EF DataSource**, with standard LINQ to Objects, and the code would be the same, only instead of **_scope.GetItems<Product>()** you would use something like **ObjectContext.Products**. In fact, we will try to do just that in a moment, once we run the project, to compare what we get from standard LINQ with what we get from **LiveLinq** here.

4. Now run the project. You see that the grid shows the filtered set of products, "expensive" products that aren't discontinued, in the order that we specified with the columns that we specified. Note also that all columns are modifiable, and you can even add and delete rows in the grid. Also note that you can sort the grid at run time by clicking column headers.

To appreciate this full data binding support, compare it with what you would get if you did not use **EF DataSource**, but if you would use standard LINQ to Objects instead. It's easy to compare, just replace **_scope.GetItems<Product>()** in the code with **App.ObjectContext.Products**. Note that you will also need to remove the type **C1.LiveLinq.LiveViews.View** and use 'var' keyword instead

for it to compile, because it will no longer be a live view. The difference is obvious: with standard LINQ, the data in the grid is read-only, and the grid does not support sorting.

But live views have even more great features. Standard LINQ to Objects produces snapshots of the data that cannot reflect changes in the source data, except some simple property changes, and even then under the strict proviso that you don't utilize a custom **Select** in your LINQ statement. Live Views, on the other hand, provide dynamic 'live' views that automatically reflect changes in their source data. As such, they simplify application development because you can, in most cases, rely on data binding to automate 'live' changes in views without the need to synchronize the changes in different parts of the application with code.

To see that the views are indeed 'live', open two forms side-by-side. Run your application and open up the Custom Columns form, which we built in Customizing View, and the Client Side Querying form, which we just built here. Make some changes to a product in the **CustomColumns** form and observe how they are reflected in the other form. If, for example, you were to increase the **UnitCost** of a product to above **30**, then it would automatically appear in the second form.

To see another example of how live views automatically synchronize themselves with changes in underlying data, follow these steps:

1. Add a live view member of the user control class:

   To write code in Visual Basic

   Visual Basic

   ```
   Private _seafoodProductsView As ClientView(Of Product)
   ```

   To write code in C#

   C#

   ```
   private ClientView<Product> _seafoodProductsView;
   ```

2. Add the following code to the form's constructor:

   To write code in Visual Basic

   Visual Basic

   ```
   _seafoodProductsView = _scope.GetItems(Of Product)().AsFiltered(Function(p)
   ```

```
p.CategoryID.Value = 8)
```

To write code in C#

C#

```
_seafoodProductsView = _scope.GetItems<Product>().AsFiltered(p => p.CategoryID == 8);
```

3. Add two buttons and the following code triggered by them:

To write code in Visual Basic

Visual Basic

```
Private Sub Button1_Click(sender As System.Object, e As
System.Windows.RoutedEventArgs)
    For Each p In _seafoodProductsView
        p.UnitPrice *= 1.2
    Next
End Sub

Private Sub Button2_Click(sender As System.Object, e As
System.Windows.RoutedEventArgs)
    For Each p In _seafoodProductsView
        p.UnitPrice /= 1.2
    Next
End Sub
```

To write code in C#

C#

```
private voidraiseSeafood(object sender, RoutedEventArgs e)

{

   foreach(var p in _seafoodProductsView)
```

```
        p.UnitPrice *= 1.2m;
    }

    private voidcutSeafood(object sender, RoutedEventArgs e)
    {
      foreach(var p in _seafoodProductsView)
        p.UnitPrice /= 1.2m;
    }
```

4. Save, build and run the application. As you press the buttons, notice how seafood products appear in the grid because their unit price is now either greater than or equal to **30** or how they disappear when their unit price falls below **30**. All of this happens automatically. You did not have to write any special code to refresh or synchronize the grid.
5. To see how live views can make almost any GUI-related code easier to write (and less error-prone), let's add a text block to the form that shows the current row count. Without **EF DataSource**, this would usually be done in a method counting the rows, and that method would have to be called in every place in the code where that count can change. In this example, there would be three such places: on initial load, and in the two methods called when the buttons are pressed, raiseSeafood and cutSeafood. So it is not very easy to synchronize display with changing data even in this simple example, not to speak of a real application. Live views make all this synchronization code unnecessary. We already saw how it works for a view containing filtering, ordering, and projection (Where, OrderBy, and Select). It works as well for views containing aggregation operations, such as Count. A little difference here is that regular LINQ Count returns a single number, to which you can't bind a control, so **EF DataSource** provides a special operation LiveCount that returns a live view instead of a single number, so you can use it in data binding. We can create this binding with a single line of code:

To write code in Visual Basic

Visual Basic

```
textBlockCount.SetBinding(TextBlock.TextProperty, New Binding("Value") With {.Source =
_viewProducts.LiveCount()})
```

To write code in C#

C#

```
textBlockCount.SetBinding(TextBlock.TextProperty, new Binding("Value") { Source =
_viewProducts.LiveCount() });
```

Simplifying MVVM

The Model-View-View-Model (**MVVM**) pattern is gaining in popularity as developers realize the benefits it gives their applications, making them easier to maintain and test and, particularly in the case of WPF and Silverlight applications, allowing a much clearer division of labor between the designer of the UI and the creator of the code that makes it work. However, without effective tools to aid program development based on **MVVM** patterns, programmers can actually find themselves working harder because of the need to implement an extra layer (the View Model) and ensure that data is properly synchronized between that and the Model. This extra burden isn't onerous when you have a relatively small application with just a few simple collections (and best practice with MVVM is to use **ObservableCollection** as the datasource), but the bigger it becomes and the more collections it spawns, the worse it becomes. **ComponentOne Entity Framework DataSource** (**EF DataSource**) can ease the burden.

EF DataSource lets you use live views as your view model. Just create live views over your model collections and use them as your view model. Live views are synchronized automatically with their sources (model), so you don't need *any* synchronization code - it's all automatic. And live views are much easier to create than to write your own view model classes using **ObservableCollection** and filling those collections manually in code. You have all the power of LINQ at your disposal to reshape model data into live views. So, not only does synchronization code disappear, but the code creating view models is dramatically simplified.

To demonstrate how easy it is to follow the **MVVM** pattern using live views, let's create a form combining all features from two previous examples: the **Category-Products** master-detail from Working with Data Sources in Code and the reshaping/filtering/ordering of data from Live Views. It will be a **Category-Products** view, showing non-discontinued products whose unit price is at least **30**, ordered by unit price, and displaying a customized set of product fields in a master-detail form where the user can select a category to show the products of that category. We'll follow the **MVVM** pattern. The form (view), called **CategoryProductsView**, only hosts GUI controls (a combo box and a grid) and does not have any code except what is used to set the data source to the view model.

All logic is in the view model class, separate from the GUI. More exactly, there are three classes: **CategoryViewModel**, **ProductViewModel**, and **CategoryProductsViewModel**. The first two are simple classes defining properties with no additional code:

To write code in Visual Basic

Visual Basic

```
Public Class CategoryProductsViewModel

    Private _scope As EntityClientScope

    Private _categories As ICollectionView
    Public Property Categories As ICollectionView
        Get
            Return _categories
        End Get
        Private Set(value As ICollectionView)
            _categories = value
        End Set
    End Property

    Private _products As ICollectionView
    Public Property Products As ICollectionView
        Get
            Return _products
        End Get
        Private Set(value As ICollectionView)
            _products = value
        End Set
    End Property

    Public Sub New()
        _scope = Application.ClientCache.CreateScope()

        Categories =
            From c In _scope.GetItems(Of Category)()
            Select New CategoryViewModel With
            {
                .CategoryID = c.CategoryID,
                .CategoryName = c.CategoryName
            }

        Products =
            From p In _scope.GetItems(Of Product)().AsFilteredBound(Function(p) p.CategoryID.Value)
                .BindFilterKey(Categories, "CurrentItem.CategoryID").Include("Supplier")
            Select New ProductViewModel With
            {
                .ProductID = p.ProductID,
                .ProductName = p.ProductName,
```

```
            .CategoryID = p.CategoryID,
            .CategoryName = p.Category.CategoryName,
            .SupplierID = p.SupplierID,
            .SupplierName = p.Supplier.CompanyName,
            .UnitPrice = p.UnitPrice,
            .QuantityPerUnit = p.QuantityPerUnit,
            .UnitsInStock = p.UnitsInStock,
            .UnitsOnOrder = p.UnitsOnOrder
        }

    End Sub
End Class
```

To write code in C#

C#

```
public class CategoryViewModel

{

    public virtual int CategoryID { get; set; }

    public virtual string CategoryName { get; set; }

}


public classProductViewModel

{

    public virtual int ProductID { get; set; }

    public virtual string ProductName { get; set; }

    public virtual int? CategoryID { get; set; }

    public virtual string CategoryName { get; set; }

    public virtual int? SupplierID { get; set; }

    public virtual string SupplierName { get; set; }
```

```
    public virtual decimal? UnitPrice { get; set; }
    public virtual string QuantityPerUnit { get; set; }
    public virtual short? UnitsInStock { get; set; }
    public virtual short? UnitsOnOrder { get; set; }
}

public classCategoryProductsViewModel
{
    private C1.Data.Entities.EntityClientScope _scope;

    public System.ComponentModel.ICollectionView Categories { get; private set; }
    public System.ComponentModel.ICollectionView Products { get; private set; }

    public CategoryProductsViewModel()
    {
        if (App.ClientCache == null)
            return;

        _scope = App.ClientCache.CreateScope();

        Categories =
            from c in _scope.GetItems<Category>()
            select new CategoryViewModel()
            {
                CategoryID = c.CategoryID,
```

```
            CategoryName = c.CategoryName
        };

    Products =
        from p in _scope.GetItems<Product>().AsFilteredBound(p => p.CategoryID)
            .BindFilterKey(Categories, "CurrentItem.CategoryID").Include("Supplier")
        select new ProductViewModel()
        {
            ProductID = p.ProductID,
            ProductName = p.ProductName,
            CategoryID = p.CategoryID,
            CategoryName = p.Category.CategoryName,
            SupplierID = p.SupplierID,
            SupplierName = p.Supplier.CompanyName,
            UnitPrice = p.UnitPrice,
            QuantityPerUnit = p.QuantityPerUnit,
            UnitsInStock = p.UnitsInStock,
            UnitsOnOrder = p.UnitsOnOrder
        };
    }
}
```

Basically, it contains just two LiveLinq statements, nothing more.

Similar to what we saw in Working with Data Sources in Code, using AsFilteredBound() gives us server-side filtering. In Working with Data Sources in Code, we connected the filter key (which is the

Product.CategoryID property) to **CategoryID** selected by the user using a combo box event. Here we can't do this because we must keep our code independent of GUI. So we use a BindFilterKey method to bind the filter key to the **Category.CategoryID** property of the item currently selected in the **Categories** collection.

The **Include**("Supplier") operator is not strictly necessary; it is used here for performance optimization. Without it, **Entity Framework** will fetch **Supplier** objects lazily, one-by-one, every time the user accesses an element of the **Products** collection whose **Supplier** was not yet fetched. This can cause delays, and it's generally much less efficient to fetch data in single rows than in batches, so we opted out of lazy loading here using **Include**("Supplier"), which tells **Entity Framework** to fetch supplier information in the same query with products.

Finally, we need to specify data binding for the GUI controls in the form (view) CategoryProductsView.xaml. Following the MVVM pattern, we do it in XAML (not in code), as shown below:

```
<Grid>

   <Grid.DataContext>

      <local:CategoryProductsViewModel/>

   </Grid.DataContext>

   <Grid.RowDefinitions>

      <RowDefinitionHeight="Auto"/>

      <RowDefinition/>

   </Grid.RowDefinitions>

   <ComboBoxHorizontalAlignment="Left"Margin="5"Name="comboBox1"
      Width="221"ItemsSource="{BindingCategories}"
      DisplayMemberPath="CategoryName" />

   <DataGridGrid.Row="1"AutoGenerateColumns="True"Name="dataGrid1"
      ItemsSource="{BindingProducts}"/>

</Grid>
```

Defining DataContext like this,

```
<Grid.DataContext>
    <local:CategoryProductsViewModel/>
</Grid.DataContext>
```

we made our form create a **CategoryProductsViewModel** object as its data source, and we bound the combo box and the grid to it using {Binding Categories} and {Binding Products}.

Using Entity Framework DataSource in MVVM with other MVVM framew

ComponentOne Entity Framework DataSource (**EF DataSource**) can be used to build Model-View-View-Model (**MVVM**) applications with any other MVVM frameworks.

ComponentOne Entity Framework DataSourceoffers several features to make your MVVM development easier:

- **Simplifies MVVM programming**

  Given that **EF DataSource** can be used to simplify MVVM programming, as seen in the Simplifying MVVM topic, it's clearly a tool that can be used for MVVM.

- **Helps create view model classes and alleviate code bloating**

  There are a plethora of tools and frameworks that developers can use to aid them in their work with MVVM, but very few that can help them create view model classes. The majority are designed to help with tasks such as passing commands and messages between the view and view model. Creating view model classes and then synchronizing them with model data is left almost entirely to manual coding. This is the primary cause of code bloating in most MVVM applications and precisely the one that **EF DataSource** is designed to alleviate in a way that is entirely compatible with other frameworks.

- **Allows you to use any framework and live views**

  You can use any framework you like to assist your MVVM application development and simply call on **EF DataSource** to provide live views to help you create view model classes.

To demonstrate these important points, we provide a sample based on the code from the well-known article by Josh Smith, one of the authors of MVVM, "WPF Apps With The Model-View-ViewModel Design Pattern" (http://msdn.microsoft.com/en-us/magazine/dd419663.aspx).

The sample can be found in the **ComponentOne Samples\Studio for WPF\C1.WPF.DataSource\CS\Orders-EF-MVVM** folder using the following paths, depending on your operating system:

Windows XP path: C:\Documents and Settings\<username>\My Documents\ComponentOne Samples

Windows 7/Vista path: C:\Users\<*username*>\Documents\.

Essentially all the files in our modified sample are the same as the originals (bar a few cosmetic changes) except one (ViewModels\OrdersViewModel.cs).

In this file, we build the view model class the **EF DataSource** way, using live views. You can see how many re-shaping functions are applied to model data to construct a view model, all done exclusively through LINQ. This made it easy and required little code. The best part is that it synchronizes automatically with model data when data in either of the two layers is changed - no synchronization code was necessary.

The fact that we only changed the way that the view model classes themselves were created (they are still derived from the original base class 'ViewModelBase') and made no other changes to the framework code that Josh Smith had employed in his original sample should serve as an example that **EF DataSource** is entirely compatible with other frameworks. You can continue to use your preferred frameworks when working with MVVM, but now you have an additional tool to make your MVVM development even easier.

Programming Guide

The following sections provide information concerning **ComponentOne Entity Framework DataSource** programming. For additional information about LiveLinq-specific features see the LiveLinq Programming Guide.

See Also

Working with Entities in Code

Using ClientView in Code

Client views are smarter than simple queries

Server-side filtering views

Working with Entities in Code

Entity Framework DataSource (EF DataSource) is non-intrusive in the sense that you can do whatever you want with entities in code using the regular Entity Framework (or RIA) methods and properties and that will work well with **EF DataSource**. You can add, delete and modify entities using the regular methods (no special **EF DataSource** object model is needed for that), and **EF DataSource** collections will automatically reflect changes that you make to the entities. For example, to add a new entity, you don't need to use some special **EF DataSource** method; therefore, none exists. Just add a new entity as you always do in EF (or RIA) and it will automatically appear in corresponding **EF DataSource** collections. If an **EF DataSource** collection has a *Where* condition, it will appear there only if it satisfies the condition. Same with deleting and modifying entities: do it the regular EF (or RIA) way, and **EF DataSource** reflects the changes automatically, so they are automatically reflected in bound controls.

However, there is one important restriction that must be strictly observed. It does not limit what you can do; it only requires you in some (relatively rare) cases to add a method call to your code notifying **EF DataSource** of what you are doing:

CAUTION

Never fetch entities from the server by directly querying the context without notifying EF DataSource.

All entities must be either fetched by one of the **EF DataSource**'s **GetItems** methods, or, if you want to fetch them by querying an object context directly, you must call the **ClientScope.AddRef** method to notify **EF DataSource** that you fetched entities from the server.

If you use the **C1DataSource** control or **ClientViewSource**, you fetch entities either implicitly if **AutoLoad** = true, or explicitly when you call the **ClientViewSource.Load** method. In both cases it is done through **EF DataSource**, so **EF DataSource** is aware of the newly fetched entities. When you

need to fetch entities in code without using ClientViewSource, the standard way to do it is by using one of the **EF DataSource**'s **GetItems** methods (EntityClientScope.GetItems|keyword=EntityClientScope.GetItems method, RiaClientScope.GetItems|keyword=RiaClientScope.GetItems method). Here too, **EF DataSource** is aware of the fetched entities. However, occasionally you may need to retrieve entities by issuing a query directly to the server without involving **EF DataSource**. For example, in Entity Framework you can take an ObjectContext that is used by **EF DataSource** and create queries:

```
ObjectContext context = scope.ClientCache.ObjectContext;
// or

ObjectQuery query = ((NORTHWNDEntities)context).Customers;
// or

query = context.CreateObjectSet<Customer>();
// or

query = context.CreateQuery<Customer>("...");
```

In RIA Services such code can look like this:

```
DomainContext context = scope.ClientCache.DomainContext;

var query = ((DomainService1)context).Customers;
// or

var entities = context.Load(

((DomainService1)context).GetCustomersQuery()).Entities;
```

Having a query, you can get entities directly from the server by enumerating the query result

```
foreach (Customer c in query) /* do something */
```

or by binding a control to it:

```
dataGrid.ItemsSource = query;
```

If you do this without calling **ClientScope.AddRef**, you can get entities from the server without **EF DataSource** knowing about it. Those entities will be in the same cache with other entities, indistinguishable from them, so **EF DataSource** will manage them as other entities, including possibly releasing them, evicting them from the cache when it does not need them. That will make the entities inaccessible, but your program may still need them! So, very bad things can happen if you fetch entities to a **EF DataSource**-governed context without using **EF DataSource** and without

notifying it that entities are fetched. Fortunately, adding that notification is easy, just call **ClientScope.AddRef** for all fetched entities like this:

```
foreach (Customer c in query)
{
   scope.AddRef(c);
   // do something
}
```

It will tell **EF DataSource** that the entities should not be released as long as the scope is alive.

Using ClientView in Code

Entity Framework DataSource (EF DataSource) supports both visual and "all code" style of programming:

- Use the **C1DataSource** control if you want point-and-click, RAD-style application development.
- Use the **ClientViewSource** class if you want to separate your code from GUI but keep the ease of use of the **C1DataSource**. The ClientViewSource class is independent of GUI; it can be used in code with any of the GUI platforms (WPF, Silverlight, WinForms). It can be used completely separately from GUI, including in the view model layer of an MVVM application. At the same time, the **ClientViewSource** is the same object **C1DataSource** uses, so you can keep the ease of use characteristic of **C1DataSource** (but code-only, without visual designers). In fact, **C1DataSource** is just a collection of **ClientViewSource** objects plus visual designer support for them.
- For the most complete control over your code, you can use the third, lowest level of the **EF DataSource** object mode: the **ClientView** class. If you prefer pure code, especially (but not only) using the MVVM pattern, this is the recommended level. It is still easy to program with, it is mostly based on LINQ which promotes a functional style of programming, intuitive and expressive.

The rest of this section is devoted to programming using the **ClientView** class.

Client views are smarter than simple queries

ClientView objects represent queries that can be executed on the server. In that regard, they provide the same functionality as queries of Entity Framework and RIA Services. When you start by creating the first ClientView in a newly created context, that's exactly what you get: an EF (or RIA) query that is executed on the server; the resulting entities are fetched to the client and made available for data binding and/or programmatic access on the client. When you continue working

on the client, that is, modify some entities on the client and query for more entities, your client views start exhibiting richer behavior that you would not get from mere EF (or RIA) queries:

- Client views are *live*. When you modify (or add, or delete) entities on the client, client views are automatically updated to reflect changed data.
- Client views make use of the **EF DataSource** client-side data cache. When you create a new client view or change an existing one, querying it does not necessarily require a round-trip to the server. **EF DataSource** looks in the cache first and uses the cache if the query can be satisfied without going to the server. That happens when a query is repeated, but not only in that case. **EF DataSource** is smart enough to detect various cases where a query can be satisfied without going to the server.

The starting point for creating client views are the **GetItems** methods of **EntityClientScope** (**RiaClientScope**):

```
ClientView<Product> products = _scope.GetItems<Product>();
```

or, in RIA Services (Silverlight):

```
ClientView<Product> products = _scope.GetItems<Product>("GetProducts");
```

(in RIA Services you need to specify the name of a query method in your domain service).

Having thus started with a base query, you can then apply filtering and paging to the view. Filtering and paging are operations that are performed on the server (if the data is not found in the cache), so applying them to a **ClientView** results in another **ClientView**. You can also apply other LINQ operators to a client view, such as grouping, sorting, and so on. That results in a **View**, which is the base class of **ClientView**. It is also a live view but it does not need ClientView functionality because those operators don't need the server; they can be performed entirely on the client.

Server-side filtering views

To apply filtering to a client view, use the **AsFiltered** method method returning **FilteredView**, a class derived from **ClientView**:

```
FilteredView<Product> productsByCategory = products.AsFiltered(p => p.CategoryID
== categoryID);
```

Here **categoryID** is a variable or parameter that is assigned a value somewhere else in the code. When you create this view, with a certain value of **categoryID**, say 1, it will retrieve products with **p.CategoryID = 1**. It will do it using the cache, without a trip to the server, if possible; if not, it will fetch those entities from the server (that's the cache-awareness feature of client views). Unlike a standard EF (or RIA) query, it will also take into account entities that are added on the client and don't yet exist on the server, and entities modified on the client so they now satisfy the filter condition (that's the *live* feature of client views).

Unlike a standard EF (or RIA) query, this view will remain up to date (live) after it was first used. If you modify data on the client and you now have more (or less) entities satisfying the condition, the view will be automatically updated to reflect the changed data. And if you have GUI controls bound to it, those will be updated too.

If you use the **AsFiltered** method with key selector function

```
FilteredView<Product> productsByCategory = products.AsFilteredBound(p =>
p.CategoryID);
```

you will be able to set/change the filter condition without creating a new view every time:

```
productsByCategory.FilterKey = categoryID;
```

There is also a convenience method, **BindFilterKey**, you can use to bind the filter key of a view to a property (or property path) of an object:

```
FilteredView<Product> productsByCategory = products.AsFilteredBound(p =>
p.CategoryID)
        .BindFilterKey(comboBox1, "SelectedValue")
```

Server-side paging views

The other server-side operation, paging, is specified with the **Paging** method returning **PagingView**, a class derived from **ClientView**:

```
PagingView<Product> productsPaged = products.Paging(p => p.ProductName, 20);
```

A paging view contains a segment (page) of data at any given time. To create a paging view, you need to specify the page size (20 in the example above) and a key selector by which to sort, because paging can only be done over sorted data.

Since a paging view is a ClientView, it supports the two fundamental features of client views: it is cache-aware and live. The performance-enhancing cache-awareness of a paging view allows it to avoid round-trips to the server when a page is requested repeatedly. And since a paging view is live (as any ClientView), it reflects changes that you make to the data on the client. For example, if you delete an entity that is currently in a paging view, it will automatically adjust itself to the changed data, including, if needed, fetching some entities from the server to occupy vacant places on the page.

Other client view operators

Progressive loading

If you have a client view that is too big to load all entities at once without a delay, you can make it load incrementally, progressively, using the **ProgressiveLoading** method:

```
ProgressiveView<Product> moreResponsiveView =
productsByCategory.ProgressiveLoading(p => p.ProductName, 100);
```
Here we intentionally used productsCategory (a **FilteredView**) instead of products, to show that any client view can be made progressive (except paging view where it does not make sense).

A progressive view loads data in portions, batches making data available on the client immediately after each portion has been loaded.

To create a progressive view, you need to specify the load size (100 in the example above) and a key selector by which to sort, because data must be sorted on the server to perform this kind of loading.

Include

Working with Entity Framework, you sometimes need to specify that related entities must be fetched together with your entity query. For example, querying for orders, you may want to get customer information for the orders to the client in the same query that fetches orders, to avoid fetching them one-by-one later (which is considerably slower than fetching all in one query). You can do this in Entity Framework by using the **Include** method of **ObjectQuery**. The same functionality is available in EF DataSource ClientView.Include method:

```
ClientView<Order> ordersWithCustomerInfo = orders.Include("Customer");
```

Live views

Client views are live; they are kept automatically up-to-date with changing data. But live views functionality is more general, live views (objects of class C1.LiveLinq.LiveViews.View from which **ClientView** is derived) can be defined on data of any kind, including but not limited to entities, and they support more LINQ query operators, such as grouping, sorting, joins and more (see ComponentOne LiveLinq).

All such operations (of which the most popular are grouping and sorting) can be applied to client views (because **ClientView** is derived from View). For example:

```
View productsByCategory = products
    .OrderBy(p => p.ProductName).GroupBy(p => p.CategoryID);
```
or in LINQ query syntax:

```
View productsByCategory =
    from p in products
    orderby p.ProductName
    group p by p.CategoryID into g
    select new { Category = g.Key, Products = g };
```
The resulting views are live, but they are not client views. This is not a defect of some sort; it is simply because such views don't need ClientView functionality. Sorting and grouping can be

performed entirely on the client without resorting to the server. However, developers must be aware of this fact to avoid confusion. Once you applied a LINQ query operation to your view, it becomes a View, not a **ClientView** (however, it remains live, automatically reflects all changes you make to the data on the client). So, for example, if you need server-side filtering, use the **AsFilter** method, not the LINQ *Where* method. Use the LINQ *Where* method when you want filtering on the client.

Client-Side Transactions

Entity Framework DataSource (EF DataSource) gives developers a powerful mechanism for canceling changes on the client without involving the server. It is called *transaction* because it is similar to the common concept of database transaction in that it allows you to ensure that a certain group of changes (unit of work) is either made in its entirety or canceled in its entirety-that your code never makes incomplete or inconsistent changes to entities in memory. It is important to understand that these transactions have no affect in any way and are completely independent from database transactions. To distinguish them from database transactions, we sometimes call them *client-side transactions*.

Client-side transactions are especially useful in implementing **Cancel/Undo** buttons/commands. Doing this without **EF DataSource** requires cancelling all changes in the object context. **EF DataSource** client-side transactions make partial canceling of changes possible. And the transactions can even be nested, so you can have, for example, a dialog box with a **Cancel** button (which cancels only the changes made in that dialog box, not all changes in the object context made elsewhere in the application), and from that dialog box you can open another dialog box with its own **Cancel** button. Using a nested (child) transaction in the child dialog ensures that its **Cancel** button cancels (rolls back) only the changes made in the child dialog box, so the user can return to editing data in the parent dialog box and then accept or cancel changes in it using the parent transaction.

The easiest way to work with client-side transactions is by associating them with live views. For example, if we have a data grid bound to a live view

```
var ordersView = from o in customer.Orders.AsLive()
                 select new OrderInfo
                 {
                     OrderID = o.OrderID,
                     OrderDate = o.OrderDate,
                 };

dataGrid1.ItemsSource = ordersView;
```

we can create a transaction and associate it with the view like this:

```
var transaction = _scope.ClientCache.CreateTransaction();
   ordersView.SetTransaction(transaction);
```

To create a child (nested) transaction, instead of calling the method **ClientCacheBase,CreateTransaction**, use the **ClientTransaction** constructor by passing it the parent transaction as a parameter:

```
var transaction = new ClientTransaction(parentTransaction);
```

Once a transaction is associated with a view by calling **View.SetTransaction**; it tracks all changes made through that view, via data binding (for example, changes made by the end user in the grid bound to the view as in the example above) as well as programmatically in code. Rolling back the transaction (calling **ClientTransaction.Rollback**) cancels the changes tracked by that transaction.

It is often necessary to bind GUI controls to a single object (as opposed to binding to a collection of objects). A single object can't be represented by a live view, so we don't have the convenience of View.SetTransaction|keyword=SetTransaction method, but in this case we can use the **ClientTransaction.ScopeDataContext** method. In WPF and Silverlight, you can use this method to set the **DataContext** so it will be used for data bindings specified in XAML:

```
DataContext = transaction.ScopeDataContext(order);
```

The resulting **DataContext** wraps the original one and performs the same data binding but with the additional benefit of all changes made through that data binding being tracked by the 'transaction', so they can be rolled back if necessary.

In WinForms, you can use the same **ScopeDataContext** and bind to the resulting object, for example, like this:

```
var dataContext = transaction.ScopeDataContext(order);

    textBox.DataBindings.Add(new Binding("Text", dataContext, "OrderDate"));
```

Finally, sometimes you need to change (or add or delete) some entities in code, not through a live view or data binding, and want those changes to be tracked by a transaction. You can do it using the **ClientTransaction.Scope()** method. That method opens a *scope* for a transaction. When you modify entities while in the scope of a transaction, those changes are tracked by that transaction. That method is designed to be used with the 'using' construct that conveniently closes the scope on exit, like this:

```
using (transaction.Scope())
{
    Customer.Orders.Add(order);
}
```

All three methods of using transactions described above are demonstrated in the **Transactions** sample project that comes with **EF DataSource**. It also demonstrates how a form with a **Cancel** button can be implemented inside another form that also has a **Cancel** button using child (nested) transactions.

Virtual Mode

Virtual mode is the technology allowing **Entity Framework DataSource (EF DataSource)** to provide data binding of GUI controls to very large data sets with no delays, no code, and without resorting to paging, with a simple setting of a property. This property is **ClientViewSource.VirtualMode**. It has three possible values:

- **None:** Virtual mode is not enabled. This is the default value.

- **Managed:** Virtual mode (fetching data from the server) is driven by a grid control. The grid control driving it is specified by setting an extender (attached) property on the grid. Most popular grids are supported, but not all grids (see the list below). If you need virtual mode with one of the supported grids, use the **Managed** option, do not use **Unmanaged**. Performance is optimal with the **Managed** option because it is specifically optimized for the grid in question. In addition to the grid control there can be other simple controls bound to the same data source (for example, TextBox controls), but no "complex" bound control (for example, another grid or list) in addition to the driving grid should be used.

Managed mode is currently supported for the following grid controls:

- **ComponentOne:** C1FlexGrid (WinForms, WPF, Silverlight), C1DataGrid (WPF, Silverlight).
- **Microsoft:** DataGridView (WinForms), DataGrid (WPF), DataGrid (Silverlight; but see below about a problem with Microsoft DataGrid for Silverlight).

- **Unmanaged:** Virtual mode (fetching data from the server) is driven by the data source itself, regardless of what type of controls are bound to it. Use this option if you need virtual mode with a grid/list/etc. control that does not belong to the list of supported grids below (but note that "simple" controls such as TextBox that are only bound to the current record and not to the entire data set can be used without limitations with either **Managed** or **Unmanaged** mode). Performance in **Unmanaged** is almost as good as in **Managed**, but it is subject to some limitations. See Unmanaged virtual mode limitations for more information. Those limitations are not very restricting, but they cannot be checked automatically, so, before deciding to use the **Unmanaged** option, make sure you do not break the limitation rules. Also, in **Unmanaged** mode, the **PageSize** property must be set to a number greater than your GUI control will show at any given time (PageSize is ignored in Managed mode).

Unmanaged virtual mode limitations

The difference between **Managed** and **Unmanaged** options is seen when data is retrieved from the server. In **Managed** mode, data is retrieved when the "driving" grid needs it, for example, when the user scrolls or navigates in the grid. In **Unmanaged** mode data is fetched from the server (if it is not already present in the cache) on every data request a bound control makes, without taking into consideration which control makes the request and for what purpose. The data source

fetches ClientViewSource.PageSize rows around every row that is requested from it. But it does not keep/hold those rows because (unlike in **Managed** mode) it does not know what rows are visible in bound controls. It holds only the current row and the last requested row (both with ClientViewSource.PageSize rows before and ClientViewSource.PageSize rows after them). All other rows are not held, that is, they can be released, evicted from cache, when **Entity Framework DataSource** needs to cleanup/compact the cache. In most common scenarios this cannot happen, and in general you can prevent it from happening by making sure that you

Never bind more than one scrollable control to a data source in Unmanaged virtual mode. You can bind any number of simple (bound to a single row) controls like TextBox. You can bind any grid or list or other "scrollable" control (bound to the entire data set as opposed to a single row) as well. But you should never bind two scrollable controls that can be scrolled to two different locations (rows) in the data set that are more than ClientViewSource.PageSize apart from one another, because only one (the latest requested) of these rows will be held, the other can be released by **Entity Framework DataSource** (at unpredictable time).

Other virtual mode limitations

- Binding two independently scrollable controls (such as grids or lists that can be scrolled to different, independent locations in the data set) to a single data source in virtual mode is not supported in **Managed** mode as well as in **Unmanaged**, and for the same reason. The difference is just that in **Managed** mode it is less dangerous because you will immediately see the effect; there is no element of unpredictability in the behavior. If in addition to the main "driving" bound control there is another scrollable control bound to the same data source, that control is not "driving", that is, scrolling in it does not fetch data; therefore, if you scroll beyond the rows that are visible in the main control, you will see empty rows.
- In both **Managed** and **Unmanaged** modes, grids, lists, and other controls bound to the entire data set (as opposed to a single record), must not do anything with all rows of the data source at once. In other words, a control should not perform actions in its code that involve going in a loop over all rows of the data source and doing something for each row. That is for the simple reason that the number of rows in virtual mode can be very large, many thousands or even millions of rows. It is, in fact, unlimited. Properly designed controls that do not rely on an assumption that the number of rows in its data source is small, will do fine in **Entity Framework DataSource (EF DataSource)** virtual mode. But those that were designed specifically for small data sources, those that loop through all their rows, can cause long delays with very large number of rows. ComponentOne C1FlexGrid and C1DataGrid (for WinForms/WPF/Silverlight) are OK, as are Microsoft DataGridView (WinForms) and Microsoft DataGrid for WPF. But Microsoft DataGrid for Silverlight (in its current version, Silverlight 4) is not recommended for **EF DataSource** virtual mode because it loops through all rows to compute its height.

ComponentOne LiveLinq

Make LINQ faster and get live views with **ComponentOne LiveLinq**™. This unique class library augments the functionality of LINQ using indexing and other optimizations to speed up LINQ queries up to 100 times faster or more. And with live view, your LINQ query result is kept up-to-date without re-populating it every time its base data changes.

See Also

LiveLinq Overview

LiveLinq in Silverlight

LiveLinq Overview

LiveLinq is a class library that augments the functionality of LINQ in two related directions:

- **It makes LINQ faster**

  LiveLinq uses indexing and other optimizations to speed up LINQ queries in memory. Speed gains can be hundreds and even thousands of times on queries with high selectivity conditions. Typical/average gains are lower but still significant; a 10-50 times speedup can be considered typical.

- **It adds support for live views to LINQ**

  A *live view* is a LINQ query result that is kept constantly up-to-date without re-populating it every time its base data changes. This makes LiveLinq extremely useful in common data-binding scenarios where objects are edited and may be filtered in or out of views, have their associated subtotals updated, and so on. In old jargon, one could say that LINQ queries correspond to snapshots, while LiveLinq views correspond to dynasets. Since live views automatically react to changes, they greatly widen the sphere of declarative programming, not only in data binding and GUI but in many other programming scenarios as well.

The scope of LiveLinq

LiveLinq implements three LINQ varieties:

- LINQ to Objects
- LINQ to XML
- LINQ to DataSet

In other words, its scope is LINQ in memory. The current LiveLinq version does not offer an alternative to LINQ to databases. However, this does not prevent you from using LiveLinq with data retrieved to memory from a database. For example, you can retrieve data from a database to a dataset in memory using ADO.NET and then use LiveLinq to DataSet. You can also retrieve objects from a database using LINQ to SQL or Entity Framework and then operate on that data using LiveLinq and send changes to the database again using Entity Framework or another framework of your choice.

LiveLinq and LINQ

LiveLinq has the same syntax as the standard LINQ. The only change necessary is to wrap your data source by applying to it an extension method **AsIndexed()** (for indexing) or **AsLive()** (for live views).

How the two parts of LiveLinq are related to each other

The two areas of LiveLinq functionality, indexing and live views can interoperate but are independent. It is not necessary to define indexes if you need to create a live view. A view is populated initially by executing a query, so, if the query can be optimized using indexing, the view will be populated faster. But after its initial population, changes to the view are made using special optimization techniques (known as Incremental View Maintenance) that don't require the user to explicitly define indexes.

Faster LINQ with Indexing

LiveLinq contains an indexing framework that it uses for optimizing query performance (not available in Silverlight). For example, by defining an index by ProductID, we can dramatically speed up queries like

```
from p in products where p.ProductID == 100
```

because it will use the index to access the requested product directly instead of going through the entire large collection in search of a single product.

Same indexes can also be used programmatically, in code, even without LINQ, for various kinds of searches, including range search and others.

Declarative programming with Live Views

Live (real) data binding

LiveLinq adds the concept of ***view*** to LINQ. A view is a query with a resultset that remains live, dynamic after it is initially populated by executing the query. A standard LINQ query is a snapshot in the sense that its result list does not change when you change the underlying (base) data. So it

can't be used for full-featured data binding. A live view is automatically kept in sync with base data, so it enables full data binding to LINQ queries.

Moreover, many views are updatable themselves; you can modify properties in their objects and add and delete objects directly in the view, see Updatable Views. So, a view can be modifiable in both directions: changes in base data propagate to the view and changes in the view propagate to base data (the latter pursuant to usual conditions of updatability).

This is full-featured two-way data binding.

Reactive = "View-Oriented" = Declarative Programming

Data binding, which is mostly used in GUI, is a very important part of live views functionality, but not the only one. More generally, live views enable a declarative style of programming which we tentatively call "view-oriented programming". Non-GUI, batch processing code can also be made declarative using live views.

Enabling technology: Incrementality

When a change occurs in base data, live views update themselves in a smart, fast way, not simply repopulating themselves from scratch. The changes are made locally, ***incrementally***, calculating the delta in the view from the delta in the base data. In most cases, it allows you to propagate the change from base data to the view very fast. This is a key component of LiveLinq: its enabling technology, based on an area of Computer Science known as Incremental View Maintenance. But the user does not need to do anything to enable this optimization, as it is done entirely beneath the hood, transparent to the user.

LiveLinq in Silverlight

LiveLinq can be used in Silverlight (starting with Silverlight 4) as well as in .NET Framework. Use the assembly

C1.Silverlight.LiveLinq.dll instead of **C1.LiveLinq.dll**.

LiveLinq indexing features are not available in Silverlight, but live views are fully supported in Silverlight as well as in .NET Framework. The following namespaces (containing indexing features and ADO.NET support unavailable in Silverlight) are not present in the Silverlight version of LiveLinq:

- C1.LiveLinq.AdoNet
- C1.LiveLinq.Indexing
- C1.LiveLinq.Indexing.Search

Two sample projects are included in Samples\LiveLinq\HowTo\LiveViews showing how to use LiveLinq in Silverlight with collections (ObservableCollection in the sample) and with XML:

- LiveViews-Silverlight-Objects
- LiveViews-Silverlight-XML

These are only samples using LiveLinq in Silverlight with in-memory objects, without database. Many more Silverlight samples using RIA Services for database access can be found in other samples, tutorials, and documentation of ComponentOne Studio for Entity Framework.

Getting Started

The following sections will help you get started with **LiveLinq**.

See Also

What is LINQ?

What is LiveLinq?

How does LiveLinq work?

What is LINQ?

LINQ, or Language Integrated Query, is a set of features in .NET 3.5 used for writing structured type-safe queries over local object collections and remote data sources.

LINQ enables you to query any collection implementing the **IEnumerable** interface, including arrays, lists, XML documents, as well as remote data sources such as tables in SQL Server.

LINQ offers the following important benefits:

- Compile-time type-checking
- Language integration (including IntelliSense support)
- Uniformity across different data sources
- Flexible, powerful, and expressive queries

In order to use **LiveLinq** effectively, you must be reasonably proficient in LINQ. A deep coverage of LINQ is beyond the scope of this document, but there are several excellent resources available on the subject. We recommend the following:

- "C# 3.0 in a nutshell", by Joseph and Ben Albahari. O'Reilly, 2007.
- "LINQ in Action", by Fabrice Marguerie, Steve Eichert and Jim Wooley. Manning, 2008.
- "Programming Microsoft LINQ Developer Reference", by Paolo Pialorsi and Marco Russo. Microsoft Press, 2008.

What is LiveLinq?

LiveLinq is a set of extensions to LINQ that add two important capabilities to standard LINQ queries:

1. **LiveLinq optimizes LINQ queries**
   LiveLinq uses indexing and other optimizations to speed up LINQ queries. Speed gains of 10 to 50 times are typical for non-trivial queries. The overall performance gains can be dramatic for data-centric applications that rely heavily on LINQ.
2. **LiveLinq turns LINQ query results into Live Views**
   Live views are query results that are kept up-to-date with respect to the base data. Live views are essential to data binding scenarios where objects may be added, removed, or changed while bound to controls in the UI. Using old jargon, one could say that plain LINQ queries correspond to *snapshots*, while **LiveLinq** views correspond to *dynasets*.

How does LiveLinq work?

LiveLinq implements "Incremental View Maintenance" techniques. Unlike standard LINQ, **LiveLinq** does not discard all query information after executing. Instead, it keeps the data it processes in indexed lists which are incrementally updated and synchronized as the underlying data changes.

The overhead involved is relatively small, since the data is already in memory to begin with (**LiveLinq** only operates with in-memory data). The benefits are huge: **LiveLinq** not only accelerates typical queries by orders of magnitude, but also enables the use of LINQ in data binding scenarios that would not be possible with standard LINQ.

Getting Started with LiveLinq

We will show several samples that illustrate **LiveLinq**. All the samples presented here use the **Northwind** database. You can use the Access version (**NWIND.MDB**) or the SQL Server version (**NORTHWND.MDF**). If you don't have Northwind and would like to try the samples, you can download the SQL Server version of the database from this URL:

http://www.microsoft.com/downloads/details.aspx?FamilyID=06616212-0356-46A0-8DA2-EEBC53A68034

The **LiveLinq** distribution package includes more samples and demos that show details not covered in the "Getting Started" part of the documentation.

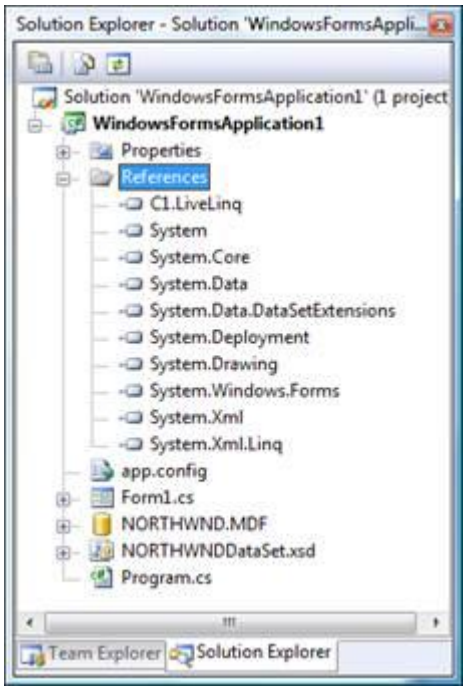
Optimizing LINQ Queries with LiveLinq

In this section, we will show a sample that illustrates how **LiveLinq** can optimize queries that use the **where** operator to filter data.

To start, follow these steps:

1. Create a new **WinForms** project
2. Add a reference to the **C1.LiveLinq.dll** assembly
3. Use the **Data | Add New DataSource** menu and add a reference to the **NORTHWND.MDF** database. Accept all the default options offered by the wizard, and pick all the tables in the database.

At this point, your project should look like this:



Next, double-click the form and add the following code:

	Copy Code

```
// declare northwind DataSet
NORTHWNDDataSet _ds = new NORTHWNDDataSet();

private void Form1_Load(object sender, EventArgs e)
{
  // load data into DataSet
  new NORTHWNDDataSetTableAdapters.CustomersTableAdapter()
    .Fill(_ds.Customers);
```

```
  new NORTHWNDDataSetTableAdapters.OrdersTableAdapter()
    .Fill(_ds.Orders);
}
```

This code declares an ADO.NET **DataSet** and loads some data into it.

Now that we have some data, let's do something with it.

Add a button to the form, double-click it, and then enter the following code:

	Copy Code

```
private void button1_Click(object sender, EventArgs e)
{
  // get reference to source data
  var customers = _ds.Customers;
  var orders = _ds.Orders;

  // find all orders for the first customer
  var q =
    from o in orders
    where o.CustomerID == customers[0].CustomerID
    select o;

  // benchmark the query (execute 1000 times)
  var start = DateTime.Now;
  int count = 0;
  for(int i = 0; i < 1000; i++)
  {
    foreach (var d in q)
      count++;
  }
  Console.WriteLine("LINQ query done in {0} ms",
     DateTime.Now.Subtract(start).TotalMilliseconds);
}
```

The code creates a simple LINQ query that enumerates all orders for the first customer in the database, then executes the query 1000 times and reports how long the process took.

If you run the project now and click the button, the Visual Studio output window should show something like this:

LINK query done in 262 ms

Now let us optimize this query with **LiveLinq**.

Start by adding the following **using** statements to the top of the code:

```
using C1.LiveLinq;

using C1.LiveLinq.AdoNet;
```

Next, add a second button to the form, double-click it and enter the following code:

	Copy Code

```
private void button2_Click(object sender, EventArgs e)
{

  // get reference to source data

  var customers = _ds.Customers;

  var orders = _ds.Orders;


  // find all orders for the first customer

  var q =

    from o in orders.AsIndexed()

    where o.CustomerID.Indexed() == customers[0].CustomerID

    select o;
```

```
  // benchmark the query (execute 1000 times)

  var start = DateTime.Now;

  int count = 0;

  for(int i = 0; i < 1000; i++)

  {

    foreach (var d in q)

      count++;

  }

  Console.WriteLine("LiveLinq query done in {0} ms",

     DateTime.Now.Subtract(start).TotalMilliseconds);

}
```

The code is almost identical to the plain LINQ version we used before. The only differences are:

- We use the **AsIndexed** method to convert the **orders** table into a **LiveLinq** indexed data source,
- We use the **Indexed** method to indicate to **LiveLinq** that it should index the data source by customer id,

AND

- The output message changed to show we are using **LiveLinq** now.

If you run the project again and click both buttons a few times, you should get something like this:

LINQ query done in 265 ms

LINQ query done in 305 ms

LINQ query done in 278 ms

LiveLinq query done in 124 ms

LiveLinq query done in 7 ms

LiveLinq query done in 2 ms

Notice how the plain LINQ query takes roughly the same amount of time whenever it executes. The **LiveLinq** query, on the other hand, is about twice as fast the first time it executes, and about **one hundred** times faster for all subsequent runs. This level of performance gain is typical for this type of query.

This happens because **LiveLinq** has to build the index when the query is executed for the first time. From then on, the same index is reused and performance improves dramatically. Note that the index is automatically maintained, so it is always up-to-date even when the underlying data changes.

In this example, the index was created when the **Indexed** method executed for the first time. You can also manage indexes using code if you need the extra control. For example, the code below declares an indexed collection and explicitly adds an index on the **CustomerID** field:

```
var ordersIndexed = _ds.Orders.AsIndexed();

ordersIndexed.Indexes.Add(o => o.CustomerID);
```

Note that the indexes are associated with the source data, and are kept even if the collection goes out of scope. If you executed the code above once, the index would remain active even after the **ordersIndexed** collection went out of scope.

Note also that the **AsIndexed** method is supported only for collections that provide change notifications (**LiveLinq** has to monitor changes in order to maintain its indexes). This requirement is satisfied for all common collections used in WinForms and WPF data binding. In particular, **AsIndexed** can be used with **DataTable**, **DataView**, **BindingList<T>**, **ObservableCollection<T>**, and LINQ to XML collections.

The **LiveLinq** installation includes a sample called **LiveLinqQueries** that contains many other examples with benchmarks, and includes queries against plain CLR objects, ADO.NET, and XML data.

Summarizing, **LiveLinq** can significantly optimize queries that search for any type of data that can be sorted. For example, searching for a specific customer or product, or listing products within a certain price range. These are typical queries that can be easily optimized to perform about a hundred times faster that they would using plain LINQ.

Basic LiveLinq Binding

Besides accelerating typical queries, **LiveLinq** also enables the use of LINQ in data binding scenarios.

To illustrate this, let us start with a simple example.

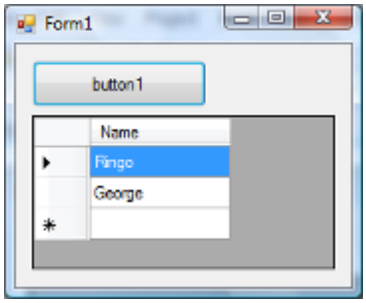
1. Create a new **WinForms** project
2. Add a reference to the **C1.LiveLinq.dll** assembly
3. Add a button and a **DataGridView** to the main form
4. Double-click the button and add this code to the form:

	Copy Code

```
using C1.LiveLinq;
using C1.LiveLinq.AdoNet;
using C1.LiveLinq.LiveViews;
using C1.LiveLinq.Collections;

private void button1_Click(object sender, EventArgs e)
{
  // create data source
  var contacts = new IndexedCollection<Contact>();

  // bind list to grid (before adding elements)
  this.dataGridView1.DataSource =
    from c in contacts.AsLive()
    where c.Name.Contains("g")
    select c;

  // add elements to collection (after binding)
  var names = "Paul,Ringo,John,George,Robert,Jimmy,John Paul,Bonzo";
  foreach (string s in names.Split(','))
  {
    contacts.Add(new Contact() { Name = s });
  }
}

public class Contact : IndexableObject
```

```
{
  private string _name;
  public string Name
  {
    get { return _name; }
    set
    {
      OnPropertyChanging("Name");
      _name = value;
      OnPropertyChanged("Name");
    }
}
```

If you run the project and click the button, you should see two names on the grid: "Ringo" and "George":



This may not seem surprising, since the data source is a query that returns all names that contain the letter "g".

The interesting part is that all the contacts were added to the list *after* the query was assigned to the grid's **DataSource** property. This shows that the **LiveLinq** query actually returned a live list, one that (a) notified the grid when elements were added to it, and (b) honored the **where** operator by showing only the names that contain "g".

The differences between this **LiveLinq** query and a regular one are:

1. The object collection is of type **IndexedCollection<T>**. This is a class provided by **LiveLinq** that supports the required change notifications.
2. The **Contact** class is derived from a **LiveLinq** class **IndexedObject** so it can provide change notifications when its properties are set.
3. The query itself contains an **AsLive** call that tells **LiveLinq** we want the result to remain active and issue change notifications that will be handled by bound controls.

You can test that the filter condition remains active by editing the grid and changing "Ringo" into "Ricky". The row will be filtered out of the view as soon as you finish editing.

You can also check the effect of the **AsLive** call by commenting it out and running the sample again. This time, the grid will not receive any notifications as items are added to the list, and will remain empty.

Hierarchical LiveLinq Binding

The previous example showed how **LiveLinq** provides basic data binding against plain CLR objects.

In this section, we will show how **LiveLinq** supports more advanced scenarios including master-detail data binding and currency management (data cursors). We will use an ADO.NET data source, but the principles are the same for all other in-memory data sources, including plain CLR objects and XML data.

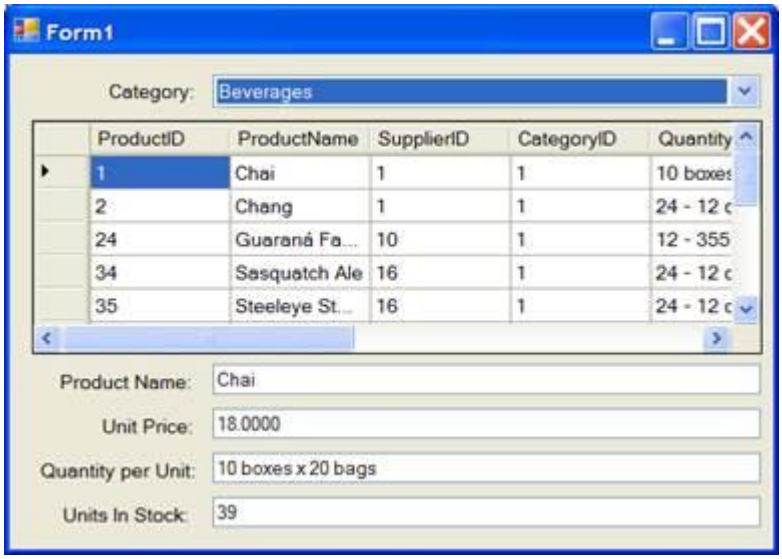
We will start with a simple application using traditional data binding. We will then show how you can easily convert that application to take advantage of **LiveLinq**. Finally, we will create WinForms and WPF versions of the application using **LiveLinq** from the start.

Traditional WinForms Implementation

Our sample application will consist of the following elements:

- A **ComboBox** listing the NorthWind product categories.
- A **DataGridView** showing all products for the currently selected category.
- Some **TextBox** controls bound to properties of the currently selected product.

This is what the final application will look like:



The basic implementation is simple, since Visual Studio handles most of the data binding related tasks automatically. In fact, the entire application can be written without a single line of code.

Here are the steps:

1. Create a new WinForms application
2. Use the **Data | Add New DataSource** menu and add a reference to the **NORTHWND.MDF** database. Accept all the default options offered by the wizard, and pick all the tables in the database.
3. Add the controls to the form as shown on the image above: one **ComboBox**, one **DataGridView**, four **TextBox** controls, and a few **Label** controls.

At this point, the application has access to the data and it has controls to show and edit the data. To connect the controls to the data, follow these steps:

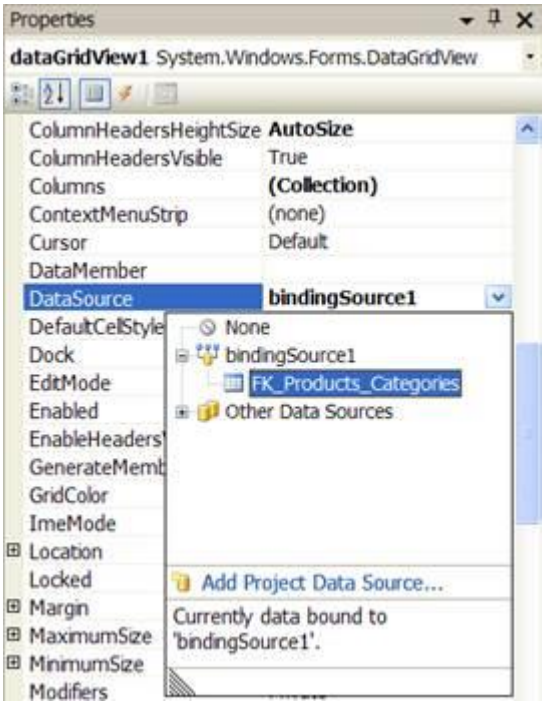
1. Add a new **BindingSource** component to the form
2. Select the new **BindingSource** component, select its **DataSource** property in the property window, and use the drop-down editor to select the **NORTHWINDDataSet** data set.
3. Still with the **BindingSource** component selected, use the drop-down editor set the **DataMember** property to "Categories".

The final step is to select each data bound control and bind it to the **BindingSource** component:

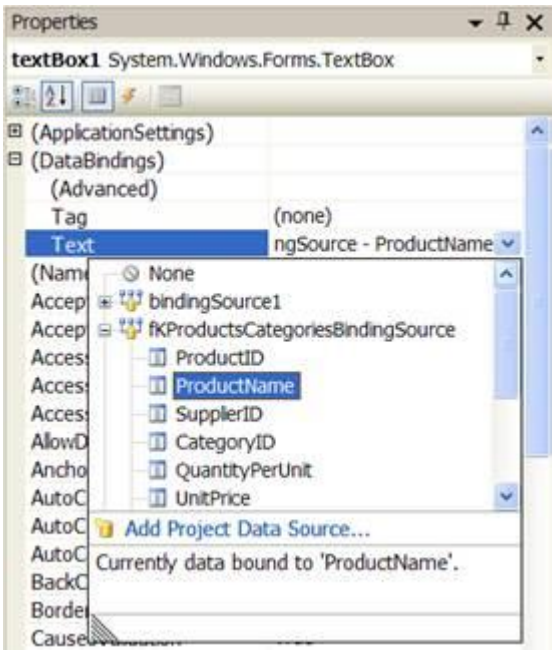
For the **ComboBox**, set the following properties:

DataSource bindingSource1
DisplayMember CategoryName
ValueMember CategoryID

For the **DataGridView**, use the drop-down editor in the property grid to set the **DataSource** property to **FK_Products_Categories**, the item that appears under bindingSource1 and represents the products under the currently selected category. The image below shows what the drop-down editor looks like just before the selection is made:



Finally, select each of the TextBox controls and use the drop-down editor in the property window to bind the Text property to the corresponding element in the currently selected product. For example:



Repeat this step to bind the other **TextBox** controls to the **UnitPrice**, **QuantityPerUnit**, and **UnitsInStock** fields.

The application is now ready. Run it and notice the following:

- When you select a category from the **ComboBox**, the corresponding products are displayed on the grid below, and the product details appear in the **TextBox** controls.
- When you select a product on the grid, the product details are automatically updated.
- The values shown on the grid and in the text boxes are synchronized. If you change the values in one place, they also change in the other.
- If you change the value of a product's **CategoryID** in the grid, the product no longer belongs to the currently selected category and is automatically removed from the grid.

This is the traditional way of doing data binding in **WinForms**. Visual Studio provides rich design-time support and tools that make it easy to get applications started. Of course, real applications typically require you to add some code to implement specific logic.

Switching to LiveLinq

The application we just described relies on a **bindingSource** object that exposes an ADO.NET **DataTable** as a data source. Migrating this application to **LiveLinq** is very easy. All you need to do is have the **bindingSource** object expose a **LiveLinq** view instead of a regular **DataTable**.

Here are the steps required:

1. Add a reference to the **C1.LiveLinq.dll** assembly to the project.
2. Add a few **using** statements to make the code more readable:

	Copy Code

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using C1.LiveLinq;
using C1.LiveLinq.AdoNet;
using C1.LiveLinq.LiveViews;
```

3. Create a **LiveLinq** query and assign it to the **DataSource** property of the **bindingSource** object, replacing the original reference to a **DataTable** object:

Example Title	Copy Code

```
private void Form1_Load(object sender, EventArgs e)
{
  // generated automatically
  this.productsTableAdapter.Fill(this.nORTHWNDDataSet.Products);
  this.categoriesTableAdapter.Fill(this.nORTHWNDDataSet.Categories);

  // Create a live view for Categories.
  // Each category contains a list with the products of that category.
  var categoryView =
    from c in nORTHWNDDataSet.Categories.AsLive()
    join p in nORTHWNDDataSet.Products.AsLive()
      on c.CategoryID equals p.CategoryID into g
    select new
    {
      c.CategoryID,
      c.CategoryName,
      FK_Products_Categories = g
    };

  // replace DataSource on the form to use our LiveLinq Query
  this.bindingSource1.DataSource = categoryView;
}
```

The code starts by creating the **LiveLinq** query that will serve as a data source for all controls on the form.

The query is 100% standard LINQ, except for the **AsLive** statements which turn the standard LINQ query into a live view suitable for binding. Without them, the code would not even compile.

The query uses a **join** to obtain all products for each category and store them in a group, and then selects the category ID, category name, and the group of products associated with the category.

The group of products is named **FK_Products_Categories**. We did not name it something simple and intuitive like "Products" because the binding code created behind the scenes by Visual Studio relies on this specific name.

If you look at the **Form1.Designer.cs** file, you will notice that Visual Studio created a **bindingSource** object named **fKProductsCategoriesBindingSource** which is initialized as follows:

Copy Code

```
//
// fKProductsCategoriesBindingSource
//
this.fKProductsCategoriesBindingSource.DataMember = "FK_Products_Categories";
this.fKProductsCategoriesBindingSource.DataSource = this.bindingSource1;
```

This code assumes that the original binding source contains a property named "FK_Products_Categories" that exposes the list of products for the current category. To ensure that the bindings still work, our query needs to use the same name.

If you run the project now, you will see that it works exactly as it did before. But now it is fully driven by a LINQ query, which means we have gained a lot of flexibility. It would be easy to rewrite the LINQ statement and display additional information such as supplier names, total sales, etc.

LiveLinq implementation in WinForms

The previous section described how to migrate a traditional data-bound application to use **LiveLinq**. The resulting application used **BindingSource** objects and code that was generated by Visual Studio at design time.

If we wanted to create a new **LiveLinq** application from scratch, we could make it even simpler.

Here are the steps:

1. Create a new WinForms application
2. Use the **Data | Add New DataSource** menu and add a reference to the **NORTHWND.MDF** database. Accept all the default options offered by the wizard, and pick all the tables in the database.
3. Add the controls to the form as before: one **ComboBox**, one **DataGridView**, four **TextBox** controls, and a few **Label** controls.
4. Add a reference to the **C1.LiveLinq.dll** assembly to the project.
5. Double-click the form add the following code:

Copy Code

```
using C1.LiveLinq;
using C1.LiveLinq.AdoNet;
using C1.LiveLinq.LiveViews;
```

```
private void Form1_Load(object sender, EventArgs e)
{
  // get the data
  var ds = GetData();

  // create a live view with Categories and Products:
  var liveView =
    from c in ds.Categories.AsLive()
    join p in ds.Products.AsLive()
      on c.CategoryID equals p.CategoryID into g
    select new
    {
      c.CategoryID,
      c.CategoryName,
      Products = g
    };

  // bind view to controls
  DataBind(liveView);
}
```

The code is straightforward. It calls a **GetData** method to load the data into a **DataSet**, creates a **LiveLinq** view using a LINQ statement similar to the one we used earlier, and then calls **DataBind** to bind the controls on the form to the view.

Here is the implementation of the **GetData** method:

	Copy Code

```
NORTHWNDDataSet GetData()
{
  NORTHWNDDataSet ds = new NORTHWNDDataSet();
  new NORTHWNDDataSetTableAdapters.ProductsTableAdapter()
    .Fill(ds.Products);
  new NORTHWNDDataSetTableAdapters.CategoriesTableAdapter()
    .Fill(ds.Categories);
  return ds;
}
```

GetData creates a **NORTHWNDDataSet**, fills the "Products" and "Categories" tables using the adapters created by Visual Studio at design time, and returns the data set.

Here is the implementation of the **DataBind** method:

	Copy Code

```
void DataBind(object dataSource)
{
  // bind ComboBox
  comboBox1.DataSource = dataSource;
  comboBox1.DisplayMember = "CategoryName";
  comboBox1.ValueMember = "CategoryID";

  // bind DataGridView
  dataGridView1.DataMember = "Products";
  dataGridView1.DataSource = dataSource;

  // bind TextBox controls
  BindTextBox(textBox1, dataSource, "ProductName");
  BindTextBox(textBox2, dataSource, "UnitPrice");
  BindTextBox(textBox3, dataSource, "QuantityPerUnit");
  BindTextBox(textBox4, dataSource, "UnitsInStock");
}

void BindTextBox(TextBox txt, object dataSource, string dataMember)
{
    var b = new Binding("Text", dataSource, "Products." + dataMember);
    txt.DataBindings.Add(b);
}
```

DataBind sets the data binding properties on each control to bind them to our **LiveLinq** view. This is exactly the same thing we did before using the property window editors, except this time we are doing it all in code and binding the controls directly to the **LiveLinq** view instead of going through a **BindingSource** component.

If you run the project now, you will see that it still works exactly as before.

Writing the data binding code usually takes a little longer than using the design time editors and letting Visual Studio write the code for you. On the other hand, the result is usually code that is simpler, easier to maintain, and easier to port to other platforms (as we will do in the next section).

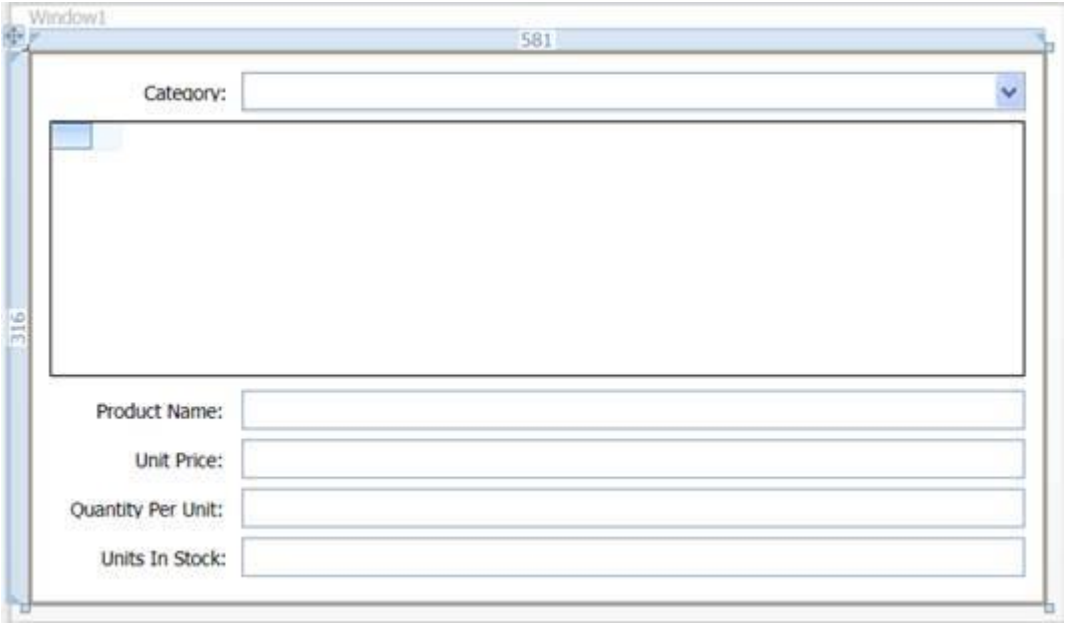
Either way, you can use **LiveLinq** views as binding sources for the controls on the form.

LiveLinq implementation in WPF

In this section, we will create another version of the same application, this time using WPF.

Here are the steps:

1. Create a new WPF application
2. Use the **Data | Add New DataSource** menu and add a reference to the **NORTHWND.MDF** database. Accept all the default options offered by the wizard, and pick all the tables in the database.
3. Add a reference to the **C1.LiveLinq.dll** assembly to the project.
4. Add a reference to a grid control such as the **C1.WPF.C1DataGrid**, which you can download from www.componentone.com.
5. Add the controls to the main window: one **ComboBox**, one **C1DataGrid**, four **TextBox** controls, and five **Label** controls. Adjust the control layout so it looks like the previous version of our application, similar to the image below:



Now right-click the window, select **View Code**, and add the following code to the project:

Copy Code

```
using System;
using System.Linq;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using C1.LiveLinq;
using C1.LiveLinq.AdoNet;
using C1.LiveLinq.LiveViews;

namespace LiveLinqWPF
{
  ///<summary>
  ///Interaction logic for Window1.xaml
  ///</summary>
  public partial class Window1 : Window
  {
    public Window1()
    {
      // designer-generated code
      InitializeComponent();

      // get data
      var ds = GetData();

      // create a live view with Categories and Products:
      var liveView =
        from c in ds.Categories.AsLive()
        join p in ds.Products.AsLive()
          on c.CategoryID equals p.CategoryID into g
        select new
        {
          CategoryID = c.CategoryID,
          CategoryName = c.CategoryName,
          Products = g
        };
```

```
      // bind view to controls
      DataBind(liveView);
    }
  }
}
```

The code is identical to the version we wrote earlier for the WinForms version of the application. It calls **GetData** to load the SQL data into a **DataSet**, then creates a **LiveLinq** view that exposes the data to the application, and finally calls the **DataBind** method to bind the controls to the **LiveLinq** view.

The **GetData** method is also identical to the one we used in the WinForms version of the application:

	Copy Code

```
NORTHWNDDataSet GetData()
{
  NORTHWNDDataSet ds = new NORTHWNDDataSet();
  new NORTHWNDDataSetTableAdapters.ProductsTableAdapter()
    .Fill(ds.Products);
  new NORTHWNDDataSetTableAdapters.CategoriesTableAdapter()
    .Fill(ds.Categories);
  return ds;
}
```

The **DataBind** method is similar to the one we wrote earlier, but it is not identical. The WPF data binding mechanism is slightly different from the one in WinForms, and that is reflected here:

	Copy Code

```
void DataBind(System.Collections.IEnumerable dataSource)
{
    // show categories in ComboBox
    comboBox1.ItemsSource = dataSource;
    comboBox1.DisplayMemberPath = "CategoryName";
```

```
    comboBox1.SelectedIndex = 0;

    // show products in C1DataGrid
    var b = new Binding("SelectedValue.Value.Products");
    b.Source = comboBox1;
    c1DataGrid1.SetBinding(C1.WPF.C1DataGrid.C1DataGrid.ItemsSourceProperty, b);

    // show product details in TextBox controls
    BindControl(textBox1, "ProductName");
    BindControl(textBox2, "UnitPrice");
    BindControl(textBox3, "QuantityPerUnit");
    BindControl(textBox4, "UnitsInStock");
}

void BindControl(TextBox txt, string dataMember)
{
    var b = new Binding("SelectedGridItem.Row.DataItem." + dataMember);
    b.Source = c1DataGrid1;
    txt.SetBinding(TextBox.TextProperty, b);
}
```

The WPF version of the **DataBind** method starts by assigning our live view to the **ItemsSource** property of the **ComboBox** control. It also sets the **DisplayMemberPath** property of the **ComboBox** to "CategoryName", which is the field we want to show in the **ComboBox**.

Next, we use the **SetBinding** method to bind the grid's **ItemsSource** property to the **ComboBox** selection. The string "SelectedValue.Value.Products" selects the "Products" field of the item that is currently selected in the **ComboBox**.

Finally, we use **SetBinding** to bind the **Text** property of each **TextBox** to the corresponding field on the grid selection. This time, we use strings like "SelectedGridItem.Row.DataItem.ProductName" which select specific properties of the product that is currently selected on the grid.

That concludes the WPF version of the application. If you run it now, you will see that it behaves exactly like the WinForms versions. Select categories from the top **ComboBox**, see the corresponding products on the grid below, and the product details in the **TextBoxes** at the bottom of the window.

LiveLinq and Declarative Programming

The live views provided by **LiveLinq** are not restricted to data binding scenarios.

Live views can be used to combine data from multiple tables, group and aggregate this data according to business rules, and make it available to the application at all times. The views are always synchronized with the data, so there's no need to call methods to update the views when you need the data. This greatly simplifies application logic and improves efficiency.

To illustrate this point, imagine a NorthWind application that exposes the following services:

ProcessOrder: This service bills the customers, ships the products, and updates the information in the database. It is used by the sales force and by the company web store.

SalesInformation: This service returns summaries of sales per product and product category. It is used by management and marketing.

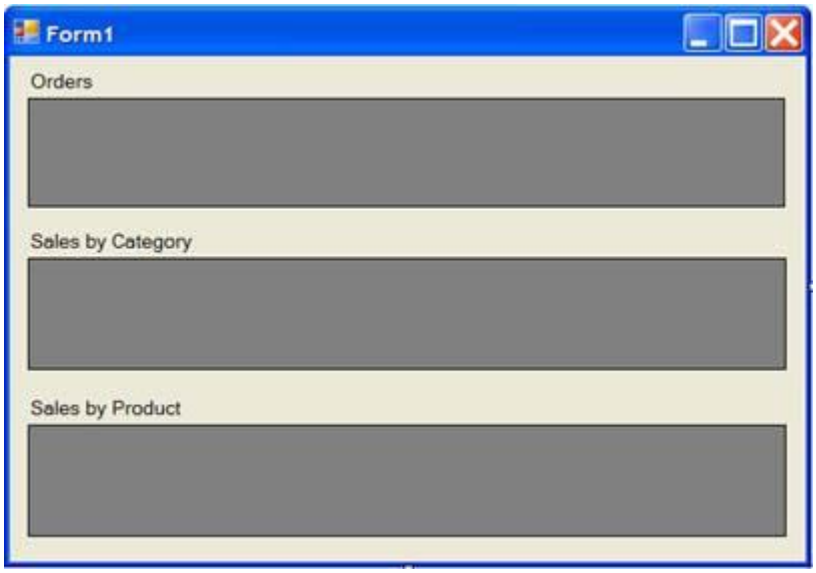
You could implement the application by having the **ProcessOrder** service write orders directly into the database and having the **SalesInformation** service run a stored procedure that would return the latest sales summaries. This would work, but the **SalesInformation** service would be relatively expensive since it would go to the database every time and would have to scan all the orders in the database.

Another approach would be to load the data into live views, where the sales summaries would be kept constantly up to date as orders are processed. Calls to the **ProcessOrder** method would automatically update the summaries provided by **SalesInformation**. Calls to **SalesInformation** would be processed in zero time, without touching the database at all.

To illustrate this, let us create another simple WinForms application using the NorthWind data once again. The application will have three grids. The first corresponds to the **ProcessOrder** service, allowing users to edit orders. The others correspond to the **SalesInformation** service, showing sales summaries that are always synchronized with the orders.

Here are the steps:

1. Create a new WinForms application
2. Use the **Data | Add New DataSource** menu and add a reference to the **NORTHWND.MDF** database. Accept all the default options offered by the wizard, and pick all the tables in the database.
3. Add a reference to the **C1.LiveLinq.dll** assembly to the project.
4. Add three **DataGridView** controls to the form, with labels above each one as shown in the image below.



Now, right-click the form and enter the following code:

	Copy Code

```
using C1.LiveLinq;
using C1.LiveLinq.AdoNet;
using C1.LiveLinq.LiveViews;

public Form1()
{
  InitializeComponent();

  // get the data
  NORTHWNDDataSet ds = GetData();

  // create a live view to update order details
  this.dataGridView1.DataSource = GetOrderDetails(ds);

  // create live views that provide up-to-date order information
  this.dataGridView2.DataSource = GetSalesByCategory(ds);
  this.dataGridView3.DataSource = GetSalesByProduct(ds);
}
```

As before, the first step is loading the relevant data from the database:

Copy Code

```
NORTHWNDDataSet GetData()
{
  var ds = new NORTHWNDDataSet();
  new NORTHWNDDataSetTableAdapters.ProductsTableAdapter()
    .Fill(ds.Products);
  new NORTHWNDDataSetTableAdapters.Order_DetailsTableAdapter()
    .Fill(ds.Order_Details);
  new NORTHWNDDataSetTableAdapters.CategoriesTableAdapter()
    .Fill(ds.Categories);
  return ds;
}
```

Next, we use **LiveLinq** to implement the live view that will be exposed through the **SalesInformation** service. This is a standard LINQ query, only slightly more sophisticated than the ones we used in earlier samples, with a couple of **AsLive** clauses that turn the standard query into a live view:

Copy Code

```
object GetSalesByCategory(NORTHWNDDataSet ds)
{
  var products = ds.Products;
  var details = ds.Order_Details;
  var categories = ds.Categories;

  var salesByCategory =
    from p in products.AsLive()
    join c in categories.AsLive()
      on p.CategoryID equals c.CategoryID
    join d in details.AsLive()
      on p.ProductID equals d.ProductID

    let detail = new
```

```
    {
      CategoryName = c.CategoryName,
      SaleAmount = d.UnitPrice * d.Quantity
        * (decimal)(1f - d.Discount)
    }

    group detail
      by detail.CategoryName into categorySales

    let total = categorySales.Sum(x => x.SaleAmount)
    orderby total descending
    select new
    {
      CategoryName = categorySales.Key,
      TotalSales = total
    };

  return salesByCategory;
}
```

The query starts by joining the three tables that contain the information on products, categories, and orders. It then creates a temporary **detail** variable that holds the product category and total amount for each order detail. Finally, the details are ordered by sales totals and grouped by category name.

The result is a live view that is automatically updated when the underlying data changes. You will see this a little later, when we run the app.

The next live view is similar, except it provides sales information by product instead of by category.

	Copy Code

```
object GetSalesByProduct(NORTHWNDDataSet ds)
{
  var products = ds.Products;
  var details = ds.Order_Details;
  var categories = ds.Categories;
```

```
  var salesByProduct =
    from p in products.AsLive()
    join c in categories.AsLive()
      on p.CategoryID equals c.CategoryID
    join d in details.AsLive()
      on p.ProductID equals d.ProductID
    into sales
    let productSales = new
    {
      ProductName = p.ProductName,
      CategoryName = c.CategoryName,
      TotalSales = sales.Sum(
         d => d.UnitPrice * d.Quantity *
              (decimal)(1f - d.Discount))
    }
    orderby productSales.TotalSales descending
    select productSales;

  return salesByProduct;
}
```

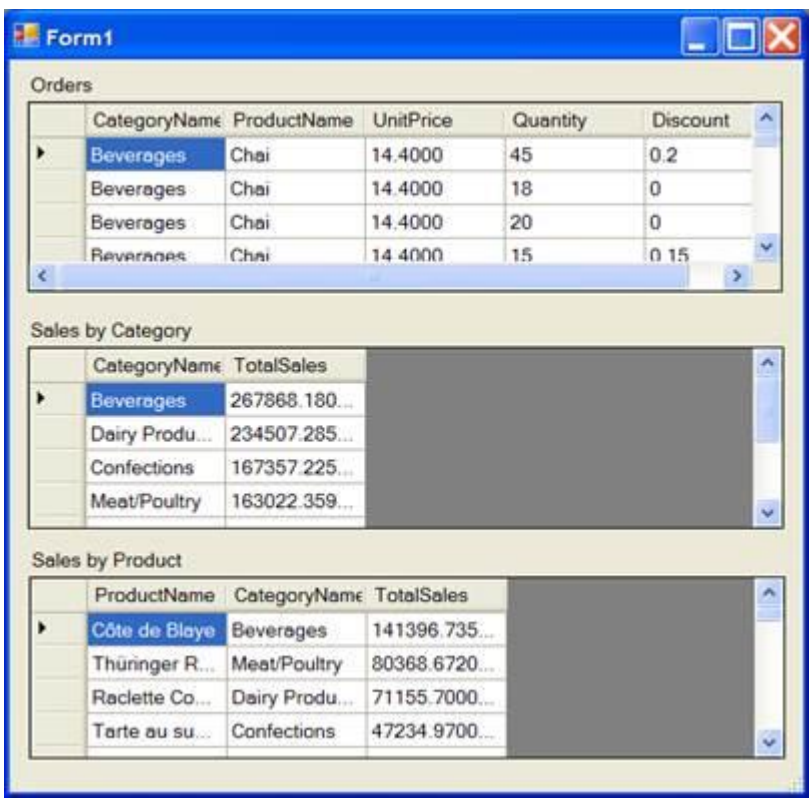
The last view shows the order details. We will bind this view to an editable grid so we can simulate orders being created or modified, and how the changes affect the previous views:

	Copy Code

```
object GetOrderDetails(NORTHWNDDataSet ds)
{
  var products = ds.Products;
  var details = ds.Order_Details;
  var categories = ds.Categories;

  var orderDetails =
    from d in details.AsLive().AsUpdatable()
    join p in products.AsLive()
      on d.ProductID equals p.ProductID
    join c in categories.AsLive()
```

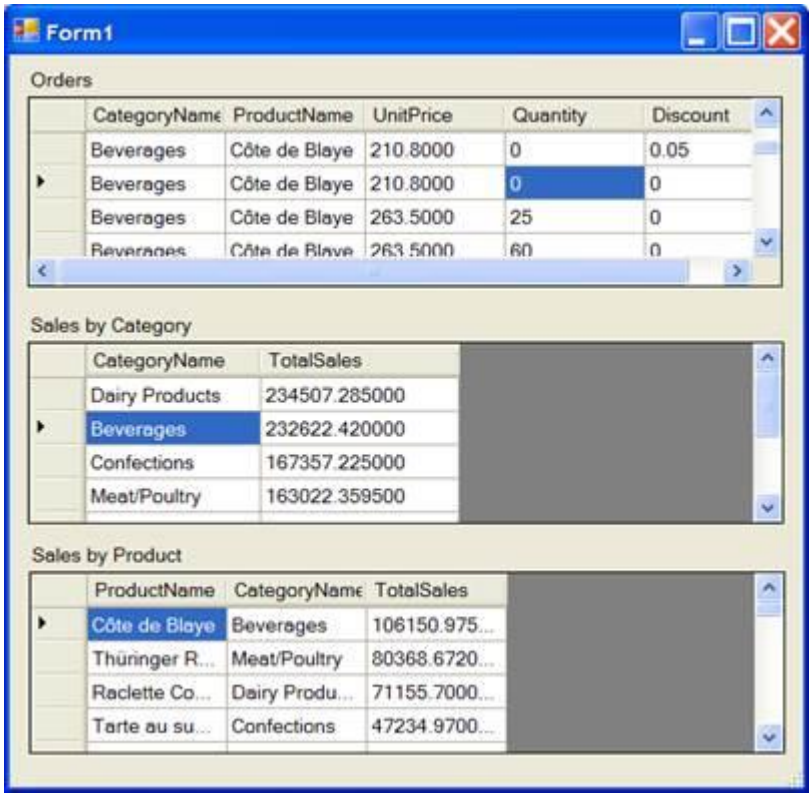
```
      on p.CategoryID equals c.CategoryID
    select new
    {
      c.CategoryName,
      p.ProductName,
      d.UnitPrice,
      d.Quantity,
      d.Discount
    };

  return orderDetails;
}
```

If you run the application now, you should see a window like this one:



The application shows the total sales by category and by product, sorted by amount. The best-selling category is “Beverages”, and the best-selling product is “Cote de Blaye”.

Now click the "ProductName" column header on the top grid and scroll down to find the entries for "Cote de Blaye". Once you've found them, try making a few changes to the orders and see how the summary data is immediately updated. For example, if you change the quantities to zero for a few "Cote de Blaye" orders, you will see that "Beverages" quickly falls behind the "Diary Products" category:



This simple example illustrates the power of LINQ-based live views. They bridge the gap between data and logic and can make data-centric applications much simpler and more efficient.

How to Use LiveLinq

How to query collections with LiveLinq

Using the built-in collection class IndexedCollection<T> (LiveLi

Using ADO.NET data collections (LiveLinq to DataSet)

Using XML data (LiveLinq to XML)

Using bindable collection classes (LiveLinq to Objects)

LiveLinq to Objects: IndexedCollection<T> and other collection c

How to create indexes

How to use indexes programmatically

How to create a live view

How to bind GUI controls to a live view

How to use live views in non-GUI code

How to query collections with LiveLinq

To use LiveLinq in Visual Studio, start by adding the LiveLinq assembly, C1.LiveLinq.dll, to your project's References. Then add the 'using' directives to the source file that will enable you to use LiveLinq in code:

	Copy Code

```
using C1.LiveLinq;
using C1.LiveLinq.Indexing;
using C1.LiveLinq.Collections;
```

(Not all of them are always necessary, but let's put all of them there at once for simplicity)

In order to query a collection in LiveLinq, we need to wrap it in an interface **IIndexedSource<T>** that will tell LiveLinq to take over. Otherwise, standard LINQ will be used. This wrapping is done with a call to the **AsIndexed** extension method, for example:

	Copy Code

```
from c in source.AsIndexed() where c.p == 1 select source
```

However, not every collection can be wrapped this way. For instance, if we try this with a **List<T>** source, we'll get a compilation error. To be usable in LiveLinq, a collection must support change notifications, it must notify LiveLinq when changes are made to its objects and when objects are added to or deleted from the collection. This requirement is satisfied for collections used for data binding, that is, implementing either **IBindingList** (WinForms data binding) or **INotifyCollectionChanged/INotifyPropertyChanged** (WPF data binding)

In particular, **AsIndexed()** is applicable to ADO.NET collections (**DataTable**, **DataView**) and to LINQ to XML collections.

Using the built-in collection class IndexedCollection<T> (LiveLinq to Objects)

Suppose first that we don't care what collection we use for our objects. We have a **Customer** class, something like

Copy Code

```
public class Customer
{
    public string Name { get; set; }
    public string City { get; set; }
}
```

and we rely on LiveLinq to supply the collection class we need.

Then we need look no further than the **C1.LiveLinq.Collections.IndexedCollection** class that is supplied by LiveLinq and is specifically optimized for LiveLinq use:

Copy Code

```
var customers = new IndexedCollection<Customer>();
customers.Add(cust1);
customers.Add(cust2);
...
var query =
  from c in customers where c.City == "London" select c;
```

Note that we can simply use **customers** instead of **customers.AsIndexed()**. It is because the **IndexedCollection<T>** class already implements the **IIndexedSource<T>** interface that LiveLinq needs, there is no need to use the **AsIndexed()** extension method to wrap it into that interface.

There is an important consideration to be kept in mind using your own classes such as **Customer** above for collection elements. The **Customer** class above is so basic that it does not support property notifications. If you set one of its properties in code, nobody will notice it, including any indexes and live views|tag=How to create a live view you may create over that collection. Therefore

it is highly necessary to provide property change notifications in such classes. Property change notifications is a standard .NET feature recommended for a variety of reasons, LiveLinq just adds another reason to do that. You can support property change notifications in your class by implementing the **INotifyPropertyChanging** and **INotifyPropertyChanged** interfaces. Or you can use LiveLinq for that, by deriving your class from IndexableObject|tag=IndexableObject Class and calling **OnPropertyChanging**/**OnPropertyChanged** like this:

	Copy Code

```
public class Customer : IndexableObject
{
    private string _name;
    public string Name
    {
        get {return _name} ;
        set
        {
            OnPropertyChanging("Name");
            _name = value;
            OnPropertyChanged("Name");
        }
    }
    private string _city;
    public string City
    {
        get {return _city};
        set
        {
            OnPropertyChanging("City");
            _city = value;
            OnPropertyChanged("City");
        }
    }
}
```

Using ADO.NET data collections (LiveLinq to DataSet)

ADO.NET **DataTable** can also be used in LiveLinq, for example:

Copy Code

```
CustomersDataTable customers = ...
var query =
  from c in customers.AsIndexed() where c.City == "London" select c;
```

For that, you'll need to add

Copy Code

```
using C1.LiveLinq.AdoNet;
```

to your source file. The **AsIndexed()** used will be C1.LiveLinq.AdoNet.AdoNetExtensions.AsIndexed. It is specifically optimized for data in ADO.NET DataSets.

Using XML data (LiveLinq to XML)

LiveLinq can query data directly from XML in memory (stored in a LINQ to XML **XDocument** class). It dramatically speeds up LINQ to XML performance by supporting XML indexing (not supported by the regular LINQ to XML).

To use LiveLinq to XML, you need to add

```
 using C1.LiveLinq.LiveViews.Xml;
```

to your source file.

Then you need to create some live views over your XML data. Unlike LiveLinq to Objects and LiveLinq to DataSet, where you can query data without live views, in LiveLinq to XML queries and live views are always used together. You are using LiveLinq to XML versions of LINQ query operators if you apply them to a live view. Otherwise, they will be the standard, not LiveLinq query operators. To start creating live views, simply apply the extension method **AsLive()** to an **XDocument**:

```
XDocument doc = …
```

```
View<XDocument> docView = doc.AsLive();
```

Once you have a live view, you can use the familiar (from LINQ to XML) query operators **Elements, Descendants** and others (see **XmlExtensions Class**) as well as all Standard Query Operators Supported in Live Views to define a live view. For example,

```
View<XElement> orders = docView.Descendants("Orders");
```

defines the collection of orders in the document. This collection is live; it is automatically kept up to date with the data in the **XDocument** that can be changed by your program. As a result, you can work with data in this collection in the same way as you would work with any dynamic collection, including ADO.NET DataTable or DataView. In particular, you can create indexes over it and use them for speeding up queries and for fast programmatic searches as shown in other sections of this documentation. It means that LiveLinq to XML makes it possible to work with XML data in memory without losing performance, so it is no longer necessary to create custom collection classes or use ADO.NET or other frameworks working with XML data. All you need is LINQ to XML with addition of LiveLinq.

Once you have some live views defined over XML data, you can query them. For example,

```
var query = from Order in orders.AsIndexed()


where (string)Order.IndexedAttribute("CustomerID") == "ALFKI"
```

gives you a query for orders of a particular customer.

Note: It is important to distinguish between live views and queries in LiveLinq to XML. Since you always start with a live view, your query will be a live view by default, unless you specify otherwise. In the example above, we used **AsIndexed()** to specify that we only need a query, don't need that query to define a live view. Live views extend query functionality, so they can be used instead of queries. However, live views have performance overhead, so you should avoid using a live view where a simple query would suffice. As a general rule, avoid creating live views and throwing them away after using them just once to get results. Live views are designed to remain active (hence they are called *live*) and show up-to-date data.

Using bindable collection classes (LiveLinq to Objects)

Any bindable collection class (a class implementing change notifications for data binding) can be used in LiveLinq queries after it is wrapped in an **IndexedCollection<T>**-derived adapter class, which is done by applying a **ToIndexed()** extension method to it.

Using **ToIndexed()**, you can apply LiveLinq to various common data collections. For example, since the **EntityCollection** and **ObjectResult** classes of the ADO.NET Entity Framework are bindable, you can use ADO.NET Entity Framework data in LiveLinq queries and views.

LiveLinq to Objects: IndexedCollection<T> and other collection classes

Querying general collection classes in LiveLinq is called LiveLinq to Objects, to distinguish it from LiveLinq to DataSet and LiveLinq to XML that query such specific data sources as DataSet and XML. So use LiveLinq to DataSet if you need to query data in a DataSet and LiveLinq to XML if your data is in XML. For other cases, we already saw two options in LiveLinq to Objects:

a. Use **IndexedCollection<T>** if you don't have preexisting collection classes and rely on LiveLinq to supply them.
b. **Apply ToIndexed()** to preexisting collection classes that support data binding.

   And there is also a more advanced option that is not frequently needed:

c. Define your own collection class, usually derived from **IndexedCollection<T>**, if you want some non-standard functionality.

   And finally, an option that can also be considered advanced, but, actually, is not particularly difficult, and allows you to bring virtually any collection into the LiveLinq orbit:

d. You can make any existing collection class **C** usable in LiveLinq provided that it somehow lets you know when changes occur. Then you can create an adapter class implementing the **C1.LiveLinq.IObservableSource<T>** interface and delegating most of its functionality to that preexisting collection class. Having an **IObservableSource<T>** implementation, you can apply the ToIndexed() extension method to it, it will return **IIndexedSource<T>**. LiveLinq extension methods implementing query operators take **IIndexedSource<T>** as their argument, see **IndexedQueryExtensions**.

How to create indexes

Now that we can query our collections in LiveLinq, we need to create some indexes for them, otherwise LiveLinq won't query them faster than the standard LINQ does.

Suppose we have a collection implementing **IIndexedSource<Customer>**, for example, like this:

```
var customers = new IndexedCollection<Customer>();
```

or like this:

```
var customers = CustomersDataTable.AsIndexed();
```

The **IIndexedSource<T>** interface has an Indexes|tag=P_C1_LiveLinq_Indexing_IIndexedSource1_Indexes collection (which is actually the only thing it has), so we use that collection to create an index like this:

```
var indexByCity = customers.Indexes.Add(x => x.City);
```

That creates an index of customers by city. Once the index is created, it will be automatically maintained on every change made to the **customers** collection. This maintenance comes with a performance cost. The cost is not high; index maintenance is fast and you can usually ignore it as long as you don't have too many indexes, but it may become a concern if you modify the collection very intensively.

To avoid heavy index maintenance cost, you can use the **BeginUpdate**/**EndUpdate** methods while making massive changes to a collection that has indexes attached to it, for example, while populating the collection.

This index will make queries such as

```
from c in customers where c.City == "London"
```

execute very fast because LiveLinq will use the index to go directly to the required items instead of searching for them by traversing the entire collection. Indexes can speed up many queries, not just this simple one, they can also speedup range conditions (with inequalities), joins, grouping, and other LINQ operators. See LiveLinq Query Performance: Tuning Indexing Performance for the full description of what classes of queries benefit from indexes.

Explicitly creating indexes by adding them to the collection as shown above is only needed if you want direct control over their lifetimes and/or direct access to the indexes (**Index<T>** objects) to use them programmatically (see How to use indexes programmatically). If all you need is to optimize a few LiveLinq queries, you can use the alternative, implicit method of creating indexes, using so called *hints* in LiveLinq queries. A hint .Indexed()|keyword=Indexed(T) Method (T) is an extension method that can be applied to a property in a query. It does not change the value of that property; it only tells LiveLinq to create an index on that property (if possible). So, instead of creating an index by city explicitly as shown above, you could write the query like this:

```
from c in customers where c.City.Indexed() == "London"
```

That would tell LiveLinq to create an index by city if it has not been already created before. This hint does not affect the value of the property, that is, the value of **c.City.Indexed()** is the same as the value of **c.City**.

How to use indexes programmatically

Indexes are used by LiveLinq to optimize query execution, but they are also accessible for programmatic use. You can use indexes directly in code, calling methods of the **Index<T>** class to perform a variety of fast searches. This makes LiveLinq indexes useful even outside the framework of LINQ, outside queries.

For example, searching for a particular value can look like this:

```
indexByCity.Find("London")
```

or even like this:

```
indexByCity.FindStartingWith("L")
```

The **Index<T>** class also has other methods for performing fast searches and fast joins and groupings that can be useful in any code, not just in queries: FindGreater, FindBetween, **Join**, **GroupJoin**, and a few others.

Examples of programmatic searches (without LINQ) can be found in the LiveLinq indexing demo, see Query Performance sample application (LiveLinqQueries). In fact, every query that is shown in that demo has an alternative implementation via direct programmatic search in code without LINQ.

How to create a live view

Consider a simple query

```
from a in A where a.p == 1 select a
```

In standard LINQ, the result of this query is a snapshot. The result collection is formed at the time when the query is executed and it does not change after that. If one of its objects changes so it no longer satisfies the condition, that object will not be removed from the result collection. And if an object in **A** changes so it now satisfies the condition, that object will not be added to the result collection. The result of even such a simple query is not live, not dynamic, does not change automatically when the base collection **A** changes, not kept in sync automatically with the base data.

In LiveLinq, if we create a view based on that query, it will be live, dynamic, will change automatically when the base data changes, will be automatically kept in sync with the base data.

All we need to do to make a view out of this query is to use the extension method **.AsLive()**:

```
var view = from a in A.AsLive() where a.p == 1 select a;
```

The **AsLive()** extension method is the analog of AsIndexed() Method/**ToIndexed()**, it can be used everywhere where the AsIndexed() Method/**ToIndexed()** extension methods can be used. So, live

views are supported in all cases in LiveLinq to Objects, LiveLinq to XML and LiveLinq to DataSet (with some restrictions on the supported query operators, see Query Operators Supported in Live Views). The difference between AsIndexed() Method/**ToIndexed()** and **AsLive()** is that **AsLive()** creates a view, thus enabling LiveLinq both to query to populate the view and to maintain the view after it has been initially populated. The AsIndexed() Method/**ToIndexed()** extension methods do only the first part, enable LiveLinq querying.

The example above is a very simple one, it's just one simple Where condition. There are other tools that can do the same, for example, **DataView** in ADO.NET or **CollectionView** in WPF. The power of LiveLinq is that it supports most of the LINQ operators, including joins and others. So you can create a live view based not just on a simple condition, but on virtually any query you need.

How to bind GUI controls to a live view

LiveLinq views can be used as data sources for GUI controls because they implement the data binding interfaces so you can simply bind any control to it as to any other data source. For example, in WinForms:

```
View<T> view = from a in A.AsLive() join b in B.AsLive() on a.k equals b.k
        where a.p == 5 select new {a, b};


dataGridView1.DataSource = view;
```

Data binding has long been the tool of choice for most developers creating GUI, but its scope was limited to simple cases only. You could bind to your base data, like your base collections, tables of a data set, etc, but not to your derived data, not to data shaped by some query. Some shaping was supported by some tools, but it was very limited, mostly just filtering and sorting. If you had data derived/shaped in any more complex way, for example, with joins (a very common case), you could not use data binding (more exactly, you could only use it for one-time snapshot binding, show data once, no changes allowed).

LiveLinq makes data binding extremely powerful by removing these limitations. Now you have the full power of LINQ to bind to.

With live views, you can create entire GUI, sophisticated applications, virtually without procedural code, using declarative data binding alone. An example of such application is the LiveLinqIssueTracker demo.

How to use live views in non-GUI code

Using live views is not limited to GUI. In a more general way, live views enable a declarative style of programming which we tentatively call ***view-oriented programming***. Non-GUI, batch processing code can also be made declarative using live views.

Generally speaking, any application operates on some data, and any data can be seen as either based or derived data. Base data contains the main objects your application is dealing with, like Customers, Orders, Employees, et cetera. Those objects (collections containing all objects of a certain class, sometimes called the *extent* of a class) are the objects that your application modifies when it needs to make some change in the base data. But applications rarely operate on this base data directly, they rarely directly operate on the entire extent of a class. Usually they filter and shape those extents and combine them together to get to a slice of data they need for a particular operation. This shaping/filtering/joining/slicing of data is the process of getting ***derived*** data (as opposed to ***base,*** underlying data). Before the emergence of LINQ, it was done by purely procedural code, with all the ensuing complexity. LINQ made it declarative, which is a big step forward. But, although declarative, it is not live, not dynamic, and does not react to changes in base data automatically. Accordingly, its reaction to change is not declarative; the programmer needs to react to changes in procedural, imperative code. Complexity is diminished by LINQ, but reacting to change remains complex. And reacting to change is pervasive because change is everywhere.

LiveLinq live views make reacting to change declarative too, thus closing the circle of declarativeness. Now entire applications, not just GUI, can be made virtually entirely declarative.

To give just a small example, we can consider a sample in the LiveLinqIssueTracker demo. It has two operations performed on some issue data:

1. Assign as many unassigned issues to employees as possible, using information on pending issues, product features (every issue belongs to a feature) and assignments.
2. Collect information on open issues for given employees.

Each of these two operations (and in a real application there is, of course, many more operations like this) works on data shaped by a query with joins (see the actual queries in Live Views How To: Use Live Views to Create Non-GUI Applications as a Set of Declarative Rules Instead of Procedural Code). Both operations can be performed more than once during program execution. Data is changing while these and other operations on data are performed. Live views used to implement these operations change automatically with that changing data, so the operations can be kept simple and completely declarative, and as a result of that, robust, reliable and flexible, easily modifiable when business requirements change.

There is nothing else needed for this style of programming in addition to what we already know about how to create live views.

Samples

Getting Started Samples

HowTo Samples

Indexing samples

Live view samples

Sample Applications

Query Performance sample application (LiveLinqQueries)

Live views sample application (LiveLinqIssueTracker)

Getting Started Samples

These are the sample projects described in the Getting Started part of the LiveLinq documentation. You can create those projects yourself following the complete step-by-step instructions in Getting Started. The pre-created projects are included in the LiveLinq distribution package for your convenience.

See Also

HowTo Samples

HowTo Samples

The following sections describe **LiveLinq**'s HowTo samples.

See Also

Indexing samples

Live view samples

Indexing samples

HowTo Indexing samples demonstrate basic use of the first of the two areas of LiveLinq functionality: using indexes to make LINQ queries faster. They don't use live views, except for LiveLinqToXml, where live views are necessary to define base data collections.

Every sample uses two queries: one the most basic and the other a little more complex, with a join. The queries are identical in functionality in all three samples, differing only in the nature of the underlying data: user object collections (LiveLinqToObjects), ADO.NET DataSet (LiveLinqDataSet) or XML (LiveLinqToXml).

Every sample shows two alternative ways of creating indexes: explicitly, by adding to the **Indexes** collection, or implicitly, by using hints in queries.

LiveLinqToObjects

This sample shows how to use the **IndexedCollection** class to create and query data collections using indexes to speed up query performance.

LiveLinqToDataSet

This sample shows how to use LiveLinq to query data residing in an ADO.NET DataSet, how to create indexes over that data that make querying faster than regular LINQ to DataSet queries. Both strongly typed and untyped datasets are demonstrated in the sample.

LiveLinqToXml

This sample shows how to use LiveLinq to query XML data. It creates indexes on 'customers' and 'orders' live views defined directly over XML data. This easy XML indexing allows to dramatically speed up LINQ queries over XML data, often make them hundreds of times faster than regular LINQ to XML, see Query Performance sample application (LiveLinqQueries) for performance metrics.

Live view samples

HowTo LiveViews samples show the second and main part of LiveLinq functionality: LINQ queries as live views, automatically reflecting changes in the data on which they are based, so they can be used to build applications declaratively, minimizing procedural, manual coding.

Every sample uses two views over the same data: one the most basic (filtered view) and the other a little more complex, with a join. The functionality is identical in all three samples, differing only in the nature of the underlying data: user object collections (LiveLinqToObjects), ADO.NET DataSet (LiveLinqDataSet) or XML (LiveLinqToXml).

To see live views in action, you can try, for example:

- Change the **ShipCity** value from "London" to "Colchester" or vice versa in any of the two grids and leave the current row (to commit the change). Observe that the value in the other grid changes accordingly.

- Change the **ShipCity** value to any string other than "London" and "Colchester" in any of the two grids. Observe that the row disappears from both grids.

See also the Live views sample application (LiveLinqIssueTracker) for more advanced functionality.

LiveViewsObjects

This sample demonstrates basic live view functionality for views based on user-defined object collections (LiveLinq to Objects).

LiveViewsDataSet

This sample demonstrates basic live view functionality for views based on ADO.NET DataSet (LiveLinq to DataSet). This sample uses a typed data set, but untyped data sets can be used as well, as shown in the LiveLinqToDataSet sample.

LiveViewsXml

This sample demonstrates basic live view functionality for views based on XML (LiveLinq to XML).

Sample Applications

The following sections describe **LiveLinq**'s sample applications.

See Also

Query Performance sample application (LiveLinqQueries)

Live views sample application (LiveLinqIssueTracker)

Query Performance sample application (LiveLinqQueries)

The querying demo shows all three LiveLinq varieties: LiveLinq to Objects, LiveLinq to DataSet and LiveLinq to XML, side by side, performing the same queries on the same data. The data are the good old Northwind **Customers** and **Orders** tables, in the three incarnations: custom object collections, a data set and XML, respectively.

The demo contains various queries, some with a parameter. For each query it shows the code, how that query is formulated in each of the three LiveLinq varieties. Every query is implemented in two different forms:

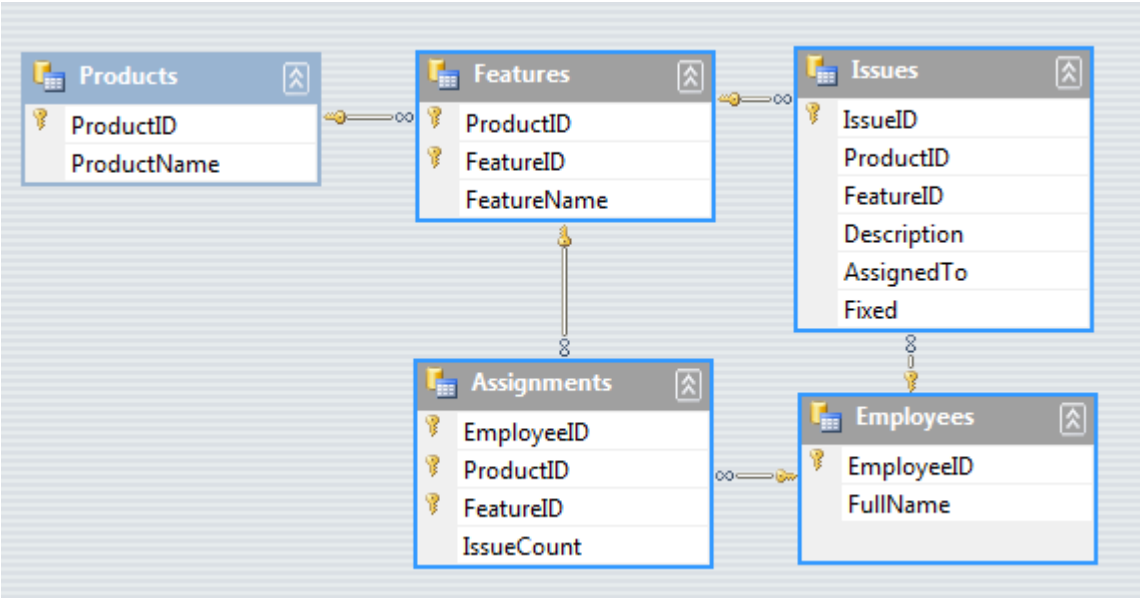
- Programmatic index search; that is, code without LINQ, directly using indexes for search, and
- LINQ query syntax.

These two LiveLinq implementations are compared with a standard LINQ implementation shown along with them. For each query, the demo shows the speedup factor: how much faster the LiveLinq implementation is compared with the standard LINQ one.

Live views sample application (LiveLinqIssueTracker)

This demo shows how live views can be used to construct entire GUI applications based almost entirely on declarative queries/views and data binding, with little procedural code. It also contains an example of how live views can be used in non-GUI, batch, perhaps server-side processing.

It is a mockup bug/issue tracking application in a fictitious company that produces Shakespeare's plays. The demo comes in three variations, each built on top of a different data store: Collections (uses LiveLinq to Objects), ADO.NET (uses LiveLinq to DataSet) and XML (uses LiveLinq to XML). It uses WinForms data binding, but WPF data binding could be used instead just as well and the live views would be exactly the same.



The data schema contains **Products** (each product is a Shakespeare's play). Each product has features (which, incidentally, are roles in the play). And there are **Issues**, in other words, bugs (such as bad acting in a certain scene). Also, there are **Employees** who are assigned those issues (incidentally, those employees are actors playing those roles), and, an important part of this schema, there are **Assignments**: every employee is assigned a certain number of features to take care of. These assignments can overlap, meaning that more than one employee can be assigned to a feature.

When we start the application and open the **Assigned Issues** form, we can see issues assigned to any given employee. The view representing it is (we are giving LiveLinq to DataSet versions of the views here, Objects and XML versions are similar):

	Copy Code

```
from i in _dataSet.Issues.AsLive()
                    join p in _dataSet.Products.AsLive()
```

```
                    on i.ProductID equals p.ProductID
                join f in _dataSet.Features.AsLive()
                    on new { i.ProductID, i.FeatureID }
                    equals new { f.ProductID, f.FeatureID }
                join e in _dataSet.Employees.AsLive()
                    on i.AssignedTo equals e.EmployeeID
                where i.AssignedTo == employeeID
                select new Issue
                {
                    IssueID = i.IssueID,
                    ProductName = p.ProductName,
                    FeatureName = f.FeatureName,
                    Description = i.Description,
                    AssignedTo = e.FullName
                };
```

It is a non-trivial LINQ query, with joins and filtering, and the demo shows that it is live, dynamic, automatically reacts to changes in data, and that GUI controls such as **DataGridView** can be bound to it.

For example, if we add an issue in the **Add Issue** form and assign it to the same employee that is shown in the **Assigned Issues** form, the list of issues for that employee is automatically updated to include the newly created issue.

The **Assigned Issues** form also demonstrates the difference between **Immediate** and **Deferred** maintenance mode, allowing switching between them with two radio buttons. If we switch to the **Deferred** mode, meaning update on demand, and add an issue, that change is not immediately reflected in the view; it is reflected only when we request data from that view, for example, by changing the employee and then changing it back.

There is also a **(Re)assign Issue** form that uses basically the same view, so, without writing code using only live views and data binding, we can do things like assign and reassign issues.

Assigned Issues and **(Re)assign Issue** forms can be opened in any number of instances side by side. You can experiment with opening several forms like that, re-assigning issues in one of them, and then observe the changes automatically reflected in all others.

The demo also shows some actions more complicated than that, such as how live views can change more than just one row at a time. There is a form called **All Issues Concerning an Employee** that shows all issues for all features assigned to a particular employee using the following view:

	Copy Code

```
from a in _dataSet.Assignments.AsLive()
                    join p in _dataSet.Products.AsLive()
                        on a.ProductID equals p.ProductID
                    join f in _dataSet.Features.AsLive()
                        on new { a.ProductID, a.FeatureID }
                            equals new { f.ProductID, f.FeatureID }
                    join i in _dataSet.Issues.AsLive()
                        on new { a.ProductID, a.FeatureID }
                            equals new { i.ProductID, i.FeatureID }
                    join e in _dataSet.Employees.AsLive()
                        on i.AssignedTo equals e.EmployeeID
                    where a.EmployeeID == employeeID
                    select new Issue
                    {
                        IssueID = i.IssueID,
                        ProductName = p.ProductName,
                        FeatureName = f.FeatureName,
                        Description = i.Description,
                        AssignedTo = e.FullName
                    };
```

Assignments of features to an employee can be changed in a **(Re)assign Features** form. You can experiment with changing the set of features assigned to an employee and see how the changes, which can be quite massive, automatically occur in the **All Issues Concerning an Employee** form. And although it's not easy to see with such a small data set (unless you add thousands of issues to it, which likely would be the case in a real-life application), the changes to the views shown in the grids are fast, without delays for re-querying the views and refreshing the entire grid, thanks to the incremental view maintenance algorithms implemented in LiveLinq.

The demo also shows how to create views based on other views and how to create indexes on views, see Live Views How To: Create Views Based on Other Views and Create Indexes on Views, and how to use live views to create non-GUI applications as a set of declarative rules instead of procedural code, see Live Views How To: Use Live Views to Create Non-GUI Applications as a Set of Declarative Rules Instead of Procedural Code.

Programming Guide

The following sections provide information concerning **LiveLinq** programming.

See Also

Query Operators Supported in Live Views

Query Expressions Supported in Live Views

View Maintenance Mode

Updatable Views

Live View Performance

LiveLinq Query Performance: logical optimization

LiveLinq Query Performance: Tuning Indexing Performance

Live Views How To: Create Views Based on Other Views and Create

Live Views How To: Use Live Views to Create Non-GUI Applications

Query Operators Supported in Live Views

LiveLinq supports all LINQ query operators in its queries. Not all operators have LiveLinq-specific implementation benefiting from indexes and other query execution optimization techniques, but such operations simply use the standard LINQ to Objects (or LINQ to XML) implementations, so it is transparent to the user.

However, not all query operations can be used in live views. This is because not all query operations have incremental view maintenance algorithms, as some would require re-populating from scratch (requerying) every time they are maintained. If your query contains such operations, you won't be able to use that query to create a live view. An attempt to do that will cause a compilation error.

Here is the list of query operators allowed in live views:

Operator	**Notes**
Select	Overload with **selector** depending on the index is not allowed.
Where	Overload with **predicate** depending on the index is not allowed.
Join	Overload with **comparer** is not allowed.

GroupJoin	Overload with **comparer** is not allowed.
OrderBy	Overload with **comparer** is not allowed.
OrderByDescending	Overload with **comparer** is not allowed.
GroupBy	Overload with **comparer** is not allowed.
SelectMany	Overloads with **selector** and **collectionSelector** depending on the index are not allowed.
Union	
Concat	
Aggregate	Use LiveAggregate method if you want to optimize performance with incremental view maintenance.
Count	Use LiveCount method if you want to optimize performance with incremental view maintenance.
Min/Max	Use LiveMin methods if you want to optimize performance with incremental view maintenance.
Sum	Use LiveSum method if you want to optimize performance with incremental view maintenance.
Average	Use LiveAverage method if you want to optimize performance with incremental view maintenance.

Query Expressions Supported in Live Views

Apart from the limitations on query operators, a query must satisfy an additional condition in order to be used as a live view. Fortunately, this condition is satisfied in most cases, so you should care about it only if your query contains some special expressions, which is relatively rare (except that all classes used in LiveLinq to Objects must satisfy the property notification condition, but that was

already mentioned in Using the built-in collection class IndexedCollection (LiveLinq to Objects). Unfortunately, this condition is not verified by LiveLinq automatically, so it is your responsibility to make sure it is satisfied. If this condition is not satisfied, your view will not react to changes in its base data. We give two descriptions of this condition: one short, for basic understanding, and another detailed, for advanced usage.

Short Description

The short answer is, basically, that in some cases your classes need to provide property notifications. There are two such cases where you need to care about that:

(1) **LiveLinq to Objects.**

There your views' arguments are collections of your own classes, so you must make sure that your classes provide property notifications, otherwise your views will not react to changes of those properties,(see Using the built-in collection class IndexedCollection<T> (LiveLinq to Objects)|tag=Using_the_built_in_collection_class_IndexedCollectionT_LiveLinq_to_Objects.).

(2) **Views over views and indexes over views** (applies to LiveLinq to DataSet and LiveLinq to XML as well as to LiveLinq to Objects).

If you have a view

View<T> v;

and want to create another view over view **v** (use **v** as its argument) or create an index over **v**, then make sure that the **T** class has property notifications, otherwise the view (or index) you define over **v** will not react to changes of those properties.

> **Note for LiveLinq to DataSet and LiveLinq to XML:** If the view result class **T** is (or is derived from) DataRow (for LiveLinq to DataSet) or XNode (for LiveLinq to XML), then property notifications there are not needed (so there are no conditions or restrictions in this case, it just works), LiveLinq gets the notifications it needs from the standard events.

Also, it can be mentioned here that you don't need to care about these conditions if the properties of your objects are not changed by your code in other ways than automatically by LiveLinq itself; that is, if you don't have code setting properties of the objects that are elements in the source collections of your views. In particular, if your classes have read-only properties, which includes all anonymous classes, that are often used in LINQ. It is also the case if you use structs, that is, value types (as opposed to classes, reference types), because value types are not references, so you can't change the elements of your collection. In all these cases there is no need in change notifications.

And finally, here is what you need to avoid: A possible source of errors is using property chains, such as for an **Order** class with a **Customer** property:

o => o.Customer.City

You can do it if you do not modify the **City** property in **Customer** objects. But if you have code changing the **City** property, **LiveLinq** will not reflect the change, because the **Order** object usually does not notify it of changes in **Customer** objects. Note that you can usually modify the query so it has the same effect without using property chains. For example, instead of

...Select(o => o.Customer.City)

use

...Select(o => o.Customer).Select(c => c.City)

Detailed Description

Observable and non-observable functions

Our condition limits the choice of functions (such as result selector, key selector, et cetera) that you can use in your query expression depending on whether or not your classes support property notifications, see Using the built-in collection class IndexedCollection (LiveLinq to Objects).

The property notifications we are talking about are those in the element classes of your view's arguments, not in the class of your view's result (but, of course, one view's result can be another view's argument, since views can be created based on views).

We distinguish two types of functions: observable and non-observable. Observable functions are allowed, non-observable functions must be avoided.

1. Observable functions. A function (expression) is observable if the only non-constant sub-expressions it contains are property or method calls whose values are observable in the sense that any change of that value is always accompanied by the change notification events.

   Examples of observable functions:

   x => x.P1

   x => x.P1 + x.P2 + x.M(c1)

   (x, y) => new {x.P1, x.P2, M = y.M(), y.P3}

   Here **P1**, **P2**, **P3**, **M** are properties/methods with change notifications, and **c1** is a value that never changes.

Note for advanced usage: Property/method call is understood here to include not only one applied directly to one of this function's parameters, but also, recursively, to the parameters of a function preceding this function in the query, if they can be reached through (possibly a chain of) simple references in object initializers. For example,

```
....Select((x, y) => new { P1 = x, P2 = y }).Select(z => new { z.P2.A, z.P1.B })
```

is allowed (if, of course, properties **A** and **B** are observable).

1a. Constants are also observable.

Note that we specifically excluded constant values from the condition above. Any constant values/objects are allowed in a function and don't break its observability. By "constant value", we mean that it remains the same object for every given parameter value. In other words, it does not change with time. The most common example of a constant is the identity function:

```
x => x
```

Although the state of the object can change with time, the object itself, as well as the reference to it, stays the same (for a given parameter object, of course, which in this case is the same as the result object), therefore it is a constant. Other examples of constants are:

```
x => c1

(x, y) => new {x, y}

(x, y) => x + y

x => x == c1 ? c2 : c3
```

where **c1**, **c2**, **c3** are some constant values.

Constant values can be combined in the same function with observable values, and the resulting function remains observable, for example:

```
x => new {x, x.P1, P = y.P2 + y.P3 }
```

where **P1** and **P2** are properties with change notifications.

1b. Object initializers and constructors that don't depend on the arguments are allowed.

The previous examples included only new with anonymous classes, but, in fact, user-defined classes and constructor calls are also allowed without breaking observability, as long as you don't use the function arguments in the constructor, using only constant expressions or nothing at all (a parameterless constructor or an object initializer). So, the

following functions are observable (where **c1** is a constant value and **P** and **Q** are observable properties):

(x, y) => new C { x.P, y.Q}

(x, y) => new C(c1) { X = x, P = y.P }

and the following are non-observable:

(x, y) => new C(y) { x.P, y.Q}

(x, y) => new C(c1, x) { x, y.P }

2. Non-observable functions.

   As the name suggests, any function that does not satisfy the condition above is considered non-observable. Note that this condition of observability depends not only on the function itself but also on the class of the argument of that function; that class must have property change notifications for everything that is used in the function that is not constant. So, even the simplest functions, such as **x => x.P** can be non-observable if **P** is non-observable (that is, the class does not issue change notifications for **P**). So, the first and most common examples of non-observable functions are the same as above but in situations where at least one of the properties **P1**, **P2** is not observable, that is, the class does not provide change notifications for it:

   x => x.P1

   (x, y) => new {x.P1, y.P2 }

   This can occur if you simply forgot to add the code providing notifications (see Using the built-in collection class IndexedCollection<T> (LiveLinq to Objects)|tag=Using_the_built_in_collection_class_IndexedCollectionT_LiveLinq_to_Objects).

   Another possible cause is a property returning a calculated value, like in

```
public class Customer
{
    public List<Order> orders;
    public int OrderCount { get {return orders.Count;} }
}
```

   unless you specifically take care to issue a property change notification in the **Customer** class every time its **orders.Count** changes.

Yet another possible cause is using a chain of properties, like, for an **Order** class with a **Customer** property:

o => o.Customer.City

Note that any expression, including the two above, can be made observable if you take special care to trigger property change notification every time its value changes, but it requires code written specifically for that purpose. For example, with the last function, you can trigger property change notification events for the **Orders** collection every time the **City** property in the **Customer** class changes.

View Maintenance Mode

Live views can be used in different kinds of applications. They can be used in GUI, interactive applications and in non-GUI, batch processing-style applications (see How to use live views in non-GUI code). Live views are optimized for both modes, GUI (interactive) and non-GUI (batch). They distinguish between these modes and operate accordingly. In GUI, live views react to a change immediately when the change happens. They do it fast, using incremental algorithms without re-populating themselves, (see Maintaining the view: Incremental View Maintenance), but still it is not always suitable for batch, non-interactive processing. In GUI, interactive programs, with data binding, immediate reaction to change is what is usually needed because the user needs to see the change on the screen. In batch processing, on the other hand, a view may be accessed long after the change occurred or even not at all. So updating that view before it is actually accessed by the program may be an unnecessary drain on resources. By default, live views distinguish between these two modes automatically, but the programmer has an option to control that using the **MaintenanceMode** property. In **Immediate** mode, which is the default for GUI, the view is maintained immediately on every change. In **Deferred** mode, which is the default in a non-GUI case (when there are no listeners attached to the view), the view is maintained on demand. It is not kept in sync with base data until it is needed or until there is a request for data from that view. When such a request arrives, the view sees that it is in a "dirty" unsynchronized state, so it automatically updates (maintains) itself to come in sync with the changed base data.

Updatable Views

Live views are bi-directionally updatable. The first direction is from base data to the view: the base (source) data can be modified and the view will be automatically updated and then synchronized with the changed base data. That is what makes the views *live*. There is also the second, opposite direction: data can be changed directly in the view. This can be done programmatically (using the ViewRow), and it can also be done via data binding. Updating data directly in the view is especially common if you bind a GUI control, such as a grid, to a view. An example of updating data in a view via data binding can be seen in one of the first samples, Basic LiveLinq Binding, among others.

We call a view *updatable* if data can be modified directly in the view. This term only concerns the second direction of updating data. The first direction mentioned above, updating base data, is always possible without restrictions for any live views. So, when we say that a view is not updatable

(is *read-only*), it does not mean that the data in the view cannot change. It can change to reflect changes in base data because every view in LiveLinq is live. It merely means that data cannot be changed directly in the view, you need to change base data if you want to modify it.

Not all live views are updatable, and even if the view as a whole is updatable, some properties of it may be read-only. These are properties that don't have direct correspondence to a property in the view's source data. Updating a view ultimately means updating that view's base data, that is, one of the view's sources, because the view itself is only virtual, does not have data of its own, shows (filtered and shaped) data of its sources. This is why a view field (property) can be updated only if it can be put into a direct correspondence with a property in the source. For example, in the following view

```
for c in customers where c.City == "London"

select new

{

   c.City,

   c.CompanyName,

   FullName = c.FirstName + " " + c.LastName

}
```

City and **CompanyName** are updatable because they directly correspond to the **City** and **CompanyName** fields of the **customers** source. But **FullName** is not updatable, because there is no single field (property) in the source to which it corresponds, it is a combination of two source properties.

Any update of a view, which can be adding, deleting or modifying a view item, is performed by performing the corresponding operation, adding, deleting or modifying a single item in one of the view's sources. Although the effect is usually obvious and it usually does what is intended, it is important to know this simple rule, to understand it formally and exactly, because the result can sometimes be unexpected if you don't take this rule into account. For example, if you modify an item (or add a new one) in the above view so its **City** value is not "London", that item will disappear from the view.

The above rule stating that updating a view item is equivalent to updating an item of one of the view's sources, implies that only one of the view's sources can be updatable. A **Join** view has two sources, so we must determine which of them is updatable. By default, a join view is read-only. If you need to make one of its two parts updatable, use the extension method AsUpdatable(), for example:

```
for o in orders.AsUpdatable()

join c in customers on o.CustomerID equals c.CustomerID

select new

{

    o.OrderDate,

    o.Amount,

    c.CompanyName

}
```

Here order data (date and amount) will be updatable and customer data (**CustomerName**) read-only.

To find out if a view is updatable, use the property View.IsReadOnly. The list of all properties of a view accessible to data binding and programmatic access, with full information including their updatability, can be obtained using the property ViewRowCollection.Properties.

Updating data directly in a view in code is done through a special class ViewRow, each view row representing a view item for programmatic access and data binding purposes. For details of using this class in code see reference documentation for ViewRowCollection and ViewRow.

Live View Performance

First performance consideration with live views is that, naturally, live view functionality has a price. Maintaining the view in sync with base data is fast and optimized, but still it consumes some resources when base data changes and the view is maintained. And it consumes some additional resources when it is first populated as well. That additional cost is not high, but it exists, manifesting itself mostly in additional memory consumption: when a live view is populated, it creates some internal data in memory to help it maintain itself faster on base data changes.

This additional memory consumption is moderate—roughly equivalent to the amount of memory needed for the resulting list itself. So, although additional resources needed for live view functionality are light to moderate, the obvious suggestion is to use live views only where you are really interested in keeping the result of a query live, or, in other words, where you need that result more than once and the base data is changing so that the result is changing too.

Other than this, there are two different aspects to live view performance:

Populating the view: query performance

Query performance is how long it takes to populate the view for the first time by executing the query (or re-populate by re-executing the query, see **View.Rebuild**). This is covered in LiveLinq query performance: logical optimization and LiveLinq Query Performance: Tuning Indexing Performance.

Maintaining the view: Incremental View Maintenance

Maintaining live views on base data changes is optimized using techniques known as Incremental View Maintenance. It means that a view does not simply re-populate itself, it adjusts to accommodate base data changes in a more smart way. The changes to the view are made locally, incrementally, calculating the delta in the view from the delta in the base data. In most cases, it is very fast and does not require any special performance tuning from the user.

As a guideline to estimating the time of view maintenance, we can say that it is roughly proportional to the number of changed items (including added and deleted ones) in the view's resulting collection. So, if only few items change as a result of a change in base data, the view maintenance time is very small and probably negligible, but it can be considerable if a considerable part of the view result changes. It can even become more costly to maintain a view than to re-query it from scratch, if, for example, half of the resulting list changes.

Finally, it is worth noting that maintenance performance for a view does not depend on its query performance. It does not matter how long it takes to populate the view initially; maintenance time is roughly the same and depends on the size of the change in (delta) and not on the size of the overall data.

LiveLinq Query Performance: logical optimization

Standard LINQ to Objects does not perform any logical optimization, it executes queries exactly as they are written. And, of course, standard LINQ to Object does not use indexes for optimization. In comparison, LiveLinq performs physical optimization (using indexes, if they are present) and logical optimization (re-writing the query in a more efficient form before processing it).

However, LiveLinq does not contain a full-scale query optimizer like one in a relational database such as SQL Server or Oracle, so it can still matter how a query is written. But it is largely limited to join ordering. LiveLinq does not try to re-order your joins. Joins are executed always in the same order as specified in the query. So, you should avoid writing queries with obviously inefficient join order. But it must be noted that it is a relatively rare problem (and the same consideration, of course, applies to the standard LINQ to Objects anyway). It can be an issue in a query like

	Copy Code

```
from a in A
join b in B on a.k equals b.k                    (1)
where b.p == 1
```

if there are, for example, only 10 elements in **B** with **b.p == 1** and only 100 elements in **A** that satisfy **a.k == b.k** and **b.p == 1**, but the overall number of elements in **A** is many times higher, for example, 10000. Then the query above will require 10000 "cycles", whereas rewritten with the right join order,

Copy Code

```
from b in B
join a in Aon b.k equals a.k                     (2)
where b.p == 1
```

this query will require only 100 cycles to complete. Note that the position of **where** is not important. The above query has the same performance as

Copy Code

```
from b in B
where b.p == 1                                    (3)
join a in A on b.k equals a.k
```

That's because of LiveLinq query optimization: LiveLinq re-writes (2) internally to (3) when it executes it, because it is preferable to check conditions before performing other operations and to exclude elements that don't contribute to the result. This is one of the logical optimizations performed by LiveLinq internally.

LiveLinq Query Performance: Tuning Indexing Performance

Query speedup achieved on any particular query by LiveLinq using indexes for optimization depends on how that query is written. This dependence is usually not dramatic. LiveLinq does a fair job of recognizing opportunities to use indexes for optimization regardless of the way the query is written. For example, it will still execute a query efficiently using indexes even if the condition consists of multiple terms connected with logical operators.

If you want to ensure that your indexes are used effectively by LiveLinq, consider the following guidelines:

Elementary predicates that can benefit from indexing

LiveLinq can use an index by a property **P** in an elementary predicate (a condition without boolean operations) that has one of the forms (patterns) (1) – (6) listed below. The simplest and perhaps the most common of such elementary predicates is a **where** condition with an equality involving a property, as in

Copy Code

```
from x in X where x.P == 1 select x
```

It will use the index by **x.P** provided that such index exists, which, as described in How to create indexes, can be ensured either by creating this index explicitly

Copy Code

```
X.Indexes.Add(x => x.P);
```

or by using the **Indexed()** method hint:

Copy Code

```
from x in X where x.P.Indexed() == 1 select x
```

In fact, LiveLinq supports a more general indexing. It does not necessarily have to be a property of **x**, it can be any expression depending on **x** (and only on **x**). This can come in handy, for example, if we are querying an untyped ADO.NET DataTable, so, because it is untyped, it does not have properties that we can index and query by their names and we need to use a query like this:

Copy Code

```
from c in customersTable.AsIndexed() where c.Field<string>("CustomerID") ==
"ALFKI" select x
```

Then we can use an index by the following expression:

	Copy Code

```
X.Indexes.Add(c => c.Field<string>("CustomerID"));
```

We will quote only the most common case of a simple property, **x.P** in the list below, but keep in mind that it can be replaced everywhere with any expression depending on **x** (and only on **x**).

Following is the list of patterns recognized by LiveLinq as allowing optimization with indexes:

1. Equality:

   x.P == Const (**Const** is a value constant in the given query operator, which means that it can be an expression depending on external parameters and variables, but it must have the same value for all elements that are tested by this **where** condition).

   Example:

	Copy Code

```
from o in Orders.AsIndexed() where o.OrderID.Indexed() == 10011
```

2. Inequality:

   x.P op Const, where **op** is one of the comparison operators: >, >=, <, <=

   Example:

	Copy Code

```
from o in Orders.AsIndexed() where o.OrderID.Indexed() > 10011
```

3. Left and right parts of an equality or inequality are interchangeable (commutative).

   Example: The following query will use indexing exactly as the one in (1):

Example Title	Copy Code

```
from o in Orders.AsIndexed() where 10011 == o.OrderID.Indexed()
```

4. StartsWith:

   x.P.StartsWith(Const), if **x.P** is of type **string**.

   Example:

	Copy Code

```
from o in Orders.AsIndexed() where o.CustomerID.StartsWith("A")
```

5. Belongs to (in, is an element of):

   ConstColl.Contains(x.P), if **ConstColl** implements **IEnumerable<T>** where **T** is the type of **x.P**.

   Example:

	Copy Code

```
from o in Orders.AsIndexed()
 where (new int[]{"ALFKI", "ANATR", "ANTON"}).Contains(o.CustomerID)
```

6. Year comparison:

   x.P.Year op Const, where **x.P** is of type **DateTime** and **op** is any of the comparison operators ==, >, >=, <, <=

   Example:

	Copy Code

```
from o in Orders.AsIndexed()
 where o.Date.Indexed().Year == 2008
```

Other elementary predicates ***don't*** use indexes. Examples where indexing will not be used:

Copy Code

```
from o in Orders.AsIndexed() where o.Freight > 10
(if there is no index defined for the Freight property)
```

Copy Code

```
from o in Orders.AsIndexed() where o.OrderID.Indexed() < o.Freight
(comparison must be with a constant, not with a variable value)
```

Copy Code

```
from o in Orders.AsIndexed() where o.OrderID.Indexed() != 10011
(only certain comparisons are used, ! (not equal) is not one of them)
```

Boolean operators

Conjunction (&&) and disjunction (||) are handled by LiveLinq optimizer, including their arbitrary combinations and including possible parentheses. Other boolean operators, such as negation (!) are not handled by the optimizer; they block its use of indexing.

Conjunction:

Conjunction does not prevent the use of indexes. For example,

Copy Code

```
from o in Orders.AsIndexed() where o.Freight > 10 && o.Lines.Count > 5
```

will use the index by **Freight** property (supposing such index exists) even though the second condition cannot use indexes. LiveLinq will simply check the other condition for each item that is found using the index from the first condition.

Conjunction of conditions using the same property

Moreover, conjunctions of conditions with the same property will be optimized to render the best execution plan. For example, if the **Freight** property is indexed,

	Copy Code

```
from o in Orders.AsIndexed() where o.Freight > 10 && o.Freight < 20
```

will not go through all orders with **Freight > 10** and check if it is less than 20 for each of them but will use the index to go directly to the orders between 10 and 20 and won't check any other orders.

Conjunction of conditions with different properties: where subindexes come into play

Conjunctions using different properties, for example

	Copy Code

```
from o in Orders.AsIndexed() where o.CustomerID == "ALFKI" && o.Freight < 20
```

can utilize subindexes, see **Subindex(T, TKey) Class**. If the index by **CustomerID** has a subindex by **Freight**, LiveLinq will not check all orders with **CustomerID** equal to "ALFKI" (which can be quite numerous). It will go directly to the subindex corresponding to the value "ALFKI" (only one such subindex exists, and it can be accessed directly, without any search) and enumerate the items in that subindex whose **Freight** is less than 20. Both operations (finding a subindex and enumerating a range of items in it) are direct access operators, without search, so the query is executed in the fastest way possible, without spending any time on checking items that don't contribute to the result.

Disjunction:

	Copy Code

```
(a)     For an indexed property x.P,

x.P == Const1 || x.P == Const2
is handled by re-writing the condition as
```

```
(new ...[]{Const1, Const2}).Contains(x.P)
or build the help system.
```

Copy Code

```
(b)     If two different properties x.P and x.Q are indexed, then

x.P op Const1 || x.Q op Const2 (op is ==, <, >, <=, etc)
is handled by re-writing the query as a union of two separate queries with
corresponding elementary conditions.
```

Join

If either of the two key selectors in a join, that is, either **x.K** or **y.K** in

Copy Code

```
from x in X
join y in Y on x.K equals y.K
```

is indexed, LiveLinq will use that index to perform the join.

Ideally, both **x.K** and **y.K** are indexed, and then LiveLinq will execute the join in the fastest way possible (using the so called *merge join* algorithm, which is very fast; basically, it takes the same time to build as to traverse the result of such join, there is no time lost on anything).

If there are no indexes on **x.K** or **y.K**, LiveLinq will still try to find the best algorithm for the join, which is sometimes merge join (if both parts of the join are ordered by the merge key) and sometimes the *hash join* algorithm used by the standard LINQ to Objects.

As a general guideline, it should be noted that defining indexes only for optimizing **Join** operations in your query will rarely lead to dramatic speedups. That's because the hash join algorithm (the one used in the standard LINQ, it does not need indexes) is already quite fast. It does make sense to define indexes for join keys sometimes, but consider it when you are fine-tuning your query, not when you need the first quick and dramatic speedup. For dramatic speedups, first consider optimizing your **Where** conditions, that can speed up your query often hundreds and even thousands times (depending on the query, of course).

Live Views How To: Create Views Based on Other Views and Create

Multiple queries with parameters can often be replaced with a single "big" live view. That can help making code clear and simple and can often make it faster. This technique is based on two features of live views:

- A view can be based on other views in the same way as it can be based on base data.
- Indexes can be created for a view in the same way as indexes are created for base data.

So, instead of issuing many queries by varying a parameter value, you can create a single live view, create an index for it, and, when you need the list of items corresponding to a particular value of the parameter, you simply get those items from the index for that particular value.

It is illustrated in the LiveLinqIssueTracker demo:

The **Assigned Issues** form uses multiple views, a separate view for every employee: views depending on a parameter **employeeID** (we are giving LiveLinq to DataSet versions of the views here, Objects and XML versions are similar):

Copy Code

```
from i in _dataSet.Issues.AsLive()
join p in _dataSet.Products.AsLive()
    on i.ProductID equals p.ProductID
join f in _dataSet.Features.AsLive()
    on new { i.ProductID, i.FeatureID }
        equals new { f.ProductID, f.FeatureID }
join e in _dataSet.Employees.AsLive()
    on i.AssignedTo equals e.EmployeeID
where i.AssignedTo == employeeID
select new Issue
{
    IssueID = i.IssueID,
    ProductName = p.ProductName,
    FeatureName = f.FeatureName,
    Description = i.Description,
    AssignedTo = e.FullName
};
```

The demo application also contains an alternative implementation of the same functionality, in the form **Assigned Issues 2**. That form uses a single view containing data for all employees, instead of multiple views depending on a parameter. This single view does not have parameters:

	Copy Code

```
_bigView =
    from i in _dataSet.Issues.AsLive()
    join p in _dataSet.Products.AsLive()
        on i.ProductID equals p.ProductID
    join f in _dataSet.Features.AsLive()
        on new { i.ProductID, i.FeatureID }
            equals new { f.ProductID, f.FeatureID }
    join e in _dataSet.Employees.AsLive()
        on i.AssignedTo equals e.EmployeeID
    select new Issue
    {
        IssueID = i.IssueID,
        ProductName = p.ProductName,
        FeatureName = f.FeatureName,
        Description = i.Description,
        AssignedToID = e.EmployeeID,
        AssignedToName = e.FullName
    };
```

That view is indexed by the employee id field:

```
 _bigView.Indexes.Add(x => x.AssignedToID);
```

so we can retrieve the items for a particular employee id at any time very fast.

Moreover, we can create a live view of that data given an employee id value (which is **comboAssignedTo.SelectedIndex** in this demo), simply by creating a view over the big view:

```
from i in _bigView where i.AssignedToID == comboAssignedTo.SelectedIndex select
i;
```

Live Views How To: Use Live Views to Create Non-GUI Applications

In addition to making GUI applications declarative, LiveLinq also enables a programming style (which we tentatively call ***view-oriented programming***) that makes non-GUI, batch processing declarative too. See How to use live views in non-GUI code for the introductory description of this technique.

The LiveLinqIssueTracker demo application contains a **Batch Processing** form where this technique is demonstrated by implementing two actions:

1. Assign as many unassigned issues to employees as possible. Looking at the feature to which a particular issue belongs, find an employee assigned to that feature, and, if that employee does not have other issues for that feature, assign that issue to that employee.
2. Collect information on all open (not fixed) issues for a given employee (perhaps for the purpose of emailing that list to that employee).

We perform action (1) by defining the following view (we are giving LiveLinq to DataSet versions of the views here, Objects and XML versions are similar):

Copy Code

```
_issuesToAssign =
    from i in issues
        where i.AssignedTo == 0
    join a in _dataSet.Assignments.AsLive()
        on new { i.ProductID, i.FeatureID }
            equals new { a.ProductID, a.FeatureID }
    join i1 in issues
        on new { i.ProductID, i.FeatureID, a.EmployeeID }
            equals new { i1.ProductID, i1.FeatureID, EmployeeID = i1.AssignedTo
}
        into g
    where g.Count() == 0
    join e in _dataSet.Employees.AsLive()
        on a.EmployeeID equals e.EmployeeID
    select new IssueAssignment
    {
        IssueID = i.IssueID,
        EmployeeID = a.EmployeeID,
        EmployeeName = e.FullName
    };
```

and performing the operation with simple code based on that view:

Copy Code

```
foreach (IssueAssignment ia in _issuesToAssign)
    _dataSet.Issues.FindByIssueID(ia.IssueID).AssignedTo = ia.EmployeeID;
```

Action (2) is performed by defining the following view:

Copy Code

```
from i in _dataSet.Issues.AsLive()
join p in _dataSet.Products.AsLive()
    on i.ProductID equals p.ProductID
join f in _dataSet.Features.AsLive()
    on new { i.ProductID, i.FeatureID }
        equals new { f.ProductID, f.FeatureID }
join em in _dataSet.Employees.AsLive()
    on i.AssignedTo equals em.EmployeeID
    where i.AssignedTo == employeeID && !i.Fixed
select i.IssueID;
```

If we need to perform the operations (1) and (2) just once, then there is no point in using live views. But suppose we are writing a program that needs to perform those actions (1) and (2) many times as steps of the overall algorithm it is executing. This is a common case, especially in server-side programming.

Without live views, we would either need to requery each time, which is very costly, or we would need to create and maintain some collections throughout our processing algorithm. Those collections would need to be maintained by manual code, often complicated, and written by different programmers and often containing bugs. Different people write different functions (actions (1) and (2) are, of course, just two small examples of such functions), they need to know what others are doing if they want to keep it consistent. All this is hard to maintain. When a new action or function is added a year after that, the logic that connects everything is by that time forgotten, the new action breaks that logic, and so the vicious cycle of too complicated programming goes on.

Compare this with how it can be done with live views:

Actions (1) and (2) are not actually procedures, they are declarative rules. Rule (1) states that this view contains the assignments to be made. Whatever changes are made to our data, this view will always contain the needed assignments, regardless of anything we add a year from now or at any other time. The logic is guaranteed to be correct because it is a declarative rule, not procedural logic.

And that rule (1) works fast. We can perform that action a thousand times over data that is always changing, and every time it will recompute only the required amount of data, only the amount that is affected by changes. It will perform only the needed ***incremental*** recomputations.

Same with rule (2): it is a declarative rule, and it executes fast without repeating calculations, recalculating only those parts that are affected by changes.

If a substantial part of our algorithm is programmed in this ***view-oriented*** way, as a set of rules like (1) and (2), then they all work together, their consistency is automatically maintained. Ideally, we can represent our entire algorithm with views/rules like that. If not, a part of it can be implemented like that and the rest can be done with the usual procedural code.

Reference

This section contains documentation for the following .NET Framework and Silverlight assemblies.

See Also

C1.Data.Entity.4 Assembly

C1.LiveLinq.4 Assembly

C1.Win.Data.Entity.4 Assembly

C1.WPF.Data.Entity.4 Assembly

C1.Silverlight.Data.Entity Assembly

C1.Silverlight.LiveLinq Assembly

C1.Data.Entity.4 Assembly

Overview

%%description%%

" -->

Namespaces

Namespace	Description
C1.Data	
C1.Data.DataSource	
C1.Data.Entities	
C1.Data.Transactions	

Namespaces

C1.Data Namespace

Overview

Classes

	Class	Description
	ClientCacheBase	Represents the client-side cache, the central hub of data access in the Studio for Entity Framework.
	ClientScope	Defines a scope of data access. Provides facilities to create client views.
	ClientView<T>	Represents a *client view*, that is, a live view that is connected to a remote source, such as an System.Data.Objects.ObjectContext.
	ClientViewLoadedEventArgs	Provides data for the ClientView<T>.Loaded event.
	DataExtensions	Extension methods provided by Studio for Entity Framework.
	FilteredView<T>	Represents a client view filtered by a filter key function, an operator and a filter key value.

	PageChangingEventArgs	Provides data for the C1.Data.DataSource.ClientCollectionView.PageChanging event.
	PagingView<T>	Represents a paged client view.
	ProgressiveView<T>	Represents a client view that loads entities sequentially (progressively) page by page. The user sees the result and can interact with it before all pages are loaded.
	SavedChangesEventArgs	Provides data for the C1DataSource.SavedChanges event.

See Also

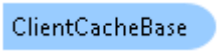
Reference

C1.Data.Entity.4 Assembly

Classes

ClientCacheBase
Represents the client-side cache, the central hub of data access in the Studio for Entity Framework.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit Class ClientCacheBase`
C#
`public abstract class ClientCacheBase`

Remarks

Usually, a single instance of this class is created on application startup with an ObjectContext/DomainContext as a parameter and exists during the entire application lifetime,

while each form, window, or user control works with data using a ClientScope created by calling the CreateScope method.

It is the base class for platform-specific implementations, such as C1.Data.Entities.EntityClientCache (Entity Framework), C1.Silverlight.Data.RiaServices.RiaClientCache (RIA Services).

Inheritance Hierarchy

System.Object
  C1.Data.ClientCacheBase
    C1.Data.Entities.EntityClientCache
    C1.Silverlight.Data.RiaServices.RiaClientCache

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

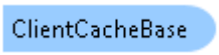
See Also

Reference

ClientCacheBase Members
C1.Data Namespace

Overview
Represents the client-side cache, the central hub of data access in the Studio for Entity Framework.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit Class ClientCacheBase`
C#
`public abstract class ClientCacheBase`

Remarks

Usually, a single instance of this class is created on application startup with an ObjectContext/DomainContext as a parameter and exists during the entire application lifetime, while each form, window, or user control works with data using a ClientScope created by calling the CreateScope method.

It is the base class for platform-specific implementations, such as C1.Data.Entities.EntityClientCache (Entity Framework), C1.Silverlight.Data.RiaServices.RiaClientCache (RIA Services).

Inheritance Hierarchy

System.Object
  C1.Data.ClientCacheBase
    C1.Data.Entities.EntityClientCache
    C1.Silverlight.Data.RiaServices.RiaClientCache

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCacheBase Members
C1.Data Namespace

Members
Methods

The following tables list the members exposed by ClientCacheBase.

Public Methods

	Name	Description
	BulkChanges	Used to group massive changes to entities and to allow manual explicit changes of entity states.
	CleanupCache	Forces unused memory to be released, unused entities to be detached from the context. It is usually done automatically, so programmers rarely need to call this

		method in code.
	Clear	Clears client-side cache entirely. Call this method if you want to make sure that following queries will fetch fresh data from the server.
	CreateScope	Creates a ClientScope that defines a scope of data access.
	CreateTransaction	Creates a C1.Data.Transactions.ClientTransaction that allows you to easily cancel changes made in transaction scope.
	Refresh	Refreshes data in all C1DataSource controls connected to this ClientCacheBase.
	RejectChanges	Reverts all pending changes for this ClientCacheBase. It is recommended to call this method instead of System.Data.Objects.ObjectContext.Refresh(System.Data.Objects.RefreshMode, System.Object).
	SaveChanges	Persists all changes to the server. It is recommended to call this method instead of System.Data.Objects.ObjectContext.SaveChanges.

Top

See Also

Reference

ClientCacheBase Class
C1.Data Namespace

Methods
For a list of all members of this type, see ClientCacheBase members.

Public Methods

	Name	Description
	BulkChanges	Used to group massive changes to entities and to allow manual explicit

		changes of entity states.
	CleanupCache	Forces unused memory to be released, unused entities to be detached from the context. It is usually done automatically, so programmers rarely need to call this method in code.
	Clear	Clears client-side cache entirely. Call this method if you want to make sure that following queries will fetch fresh data from the server.
	CreateScope	Creates a ClientScope that defines a scope of data access.
	CreateTransaction	Creates a C1.Data.Transactions.ClientTransaction that allows you to easily cancel changes made in transaction scope.
	Refresh	Refreshes data in all C1DataSource controls connected to this ClientCacheBase.
	RejectChanges	Reverts all pending changes for this ClientCacheBase. It is recommended to call this method instead of System.Data.Objects.ObjectContext.Refresh(System.Data.Objects.RefreshMode, System.Object).
	SaveChanges	Persists all changes to the server. It is recommended to call this method instead of System.Data.Objects.ObjectContext.SaveChanges.

Top

See Also

Reference

ClientCacheBase Class
C1.Data Namespace

BulkChanges Method
Example

A delegate that makes changes in entities.

Used to group massive changes to entities and to allow manual explicit changes of entity states.

Syntax

Visual Basic (Declaration)

```
Public Sub BulkChanges( _
   ByVal makeChanges As Action _
)
```

C#

```
public void BulkChanges(
   Action makeChanges
)
```

Parameters

makeChanges

A delegate that makes changes in entities.

Remarks

Internal state of the client-side cache and all existing client views based on the cache are kept unchanged, aren't updated while the given makeChanges is executed. After the delegate completes its execution (having modified multiple entities), the client-side cache internal state is restored and client views are updated (maintained) to reflect the changes made in entities during the delegate's execution.

There are two main scenarios where you should consider calling this method:

1. Using this method when you make a lot of changes to entities can improve performance because the change processing is deferred, occurs only once after all changes are done instead of every time on each change. Depending on the amount of changes, the speedup can be considerable.
2. You must use this method when you need to make changes to entity states by calling any of the following methods:
   - System.Data.Objects.ObjectStateEntry.ChangeState/SetModified/AcceptChanges,
   - System.Data.Objects.ObjectContext.AcceptAllChanges,
   - System.ServiceModel.DomainServices.Client.DomainContext.RejectChanges,
   - System.ServiceModel.DomainServices.Client.Entity.AcceptChanges/RejectChanges,
   - System.ServiceModel.DomainServices.Client.EntitySet.AcceptChanges/RejectChanges,

- System.Windows.Controls.DomainDataSource.RejectChanges.

Calling these methods without wrapping them with BulkChanges can corrupt the client-side cache.

Example

- C#

```
var scope = clientCache.CreateScope();
clientCache.BulkChanges(delegate {
    foreach(var detail in scope.GetItems<Order_Details>)
        detail.Discount *= 2;
});
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCacheBase Class
ClientCacheBase Members

CleanupCache Method

Forces unused memory to be released, unused entities to be detached from the context. It is usually done automatically, so programmers rarely need to call this method in code.

Syntax

Visual Basic (Declaration)
`Public Sub CleanupCache()`
C#
`public void CleanupCache()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCacheBase Class
ClientCacheBase Members

Clear Method
Clears client-side cache entirely. Call this method if you want to make sure that following queries will fetch fresh data from the server.

Syntax

Visual Basic (Declaration)
`Public Sub Clear()`
C#
`public void Clear()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCacheBase Class
ClientCacheBase Members

CreateScope Method
Creates a ClientScope that defines a scope of data access.

Syntax

Visual Basic (Declaration)
`Public Function CreateScope() As ClientScope`
C#

```
public ClientScope CreateScope()
```

Return Value

A new ClientScope.

Remarks

Usually, every form, window, or user control creates a ClientScope and uses it to access entities.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCacheBase Class
ClientCacheBase Members
ClientScope Class

CreateTransaction Method
Creates a C1.Data.Transactions.ClientTransaction that allows you to easily cancel changes made in transaction scope.

Syntax

Visual Basic (Declaration)
`Public Function CreateTransaction() As ClientTransaction`
C#
`public ClientTransaction CreateTransaction()`

Return Value

A new C1.Data.Transactions.ClientTransaction

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCacheBase Class
ClientCacheBase Members

Refresh Method
Refreshes data in all C1DataSource controls connected to this ClientCacheBase.

Syntax

Visual Basic (Declaration)
`Public Sub Refresh()`
C#
`public void Refresh()`

Remarks

This method calls C1DataSource.Refresh() for every C1DataSource connected to this ClientCacheBase. Changes made on the client are preserved.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCacheBase Class
ClientCacheBase Members

RejectChanges Method
Reverts all pending changes for this ClientCacheBase. It is recommended to call this method instead of System.Data.Objects.ObjectContext.Refresh(System.Data.Objects.RefreshMode,System.Object).

Syntax

Visual Basic (Declaration)
`Public Sub RejectChanges()`
C#
`public void RejectChanges()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCacheBase Class
ClientCacheBase Members

SaveChanges Method
Persists all changes to the server. It is recommended to call this method instead of System.Data.Objects.ObjectContext.SaveChanges.

Syntax

Visual Basic (Declaration)
`Public Sub SaveChanges()`
C#
`public void SaveChanges()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

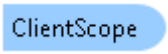
See Also

Reference

ClientCacheBase Class
ClientCacheBase Members

ClientScope
Defines a scope of data access. Provides facilities to create client views.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class ClientScope`
C#
`public class ClientScope`

Remarks

Usually, one scope is created per form/window. Entities pinned to the scope (marked as needed) are not evicted from the cache until the scope is disposed or collected by the garbage collector.

This class is a base class for platform-specific scopes, such as C1.Data.Entities.EntityClientCache (Entity Framework) and C1.Silverlight.Data.RiaServices.RiaClientCache (RIA Services).

Inheritance Hierarchy

System.Object
  C1.Data.ClientScope
    C1.Data.Entities.EntityClientScope
    C1.Silverlight.Data.RiaServices.RiaClientScope

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

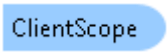
See Also

Reference

ClientScope Members
C1.Data Namespace

Overview
Defines a scope of data access. Provides facilities to create client views.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class ClientScope`
C#
`public class ClientScope`

Remarks

Usually, one scope is created per form/window. Entities pinned to the scope (marked as needed) are not evicted from the cache until the scope is disposed or collected by the garbage collector.

This class is a base class for platform-specific scopes, such as C1.Data.Entities.EntityClientCache (Entity Framework) and C1.Silverlight.Data.RiaServices.RiaClientCache (RIA Services).

Inheritance Hierarchy

System.Object
  C1.Data.ClientScope
    C1.Data.Entities.EntityClientScope
    C1.Silverlight.Data.RiaServices.RiaClientScope

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientScope Members
C1.Data Namespace

Members
Properties Methods

The following tables list the members exposed by ClientScope.

Public Constructors

	Name	Description
	ClientScope Constructor	Initializes a new instance of ClientScope class with the given ClientCacheBase.

Top

Public Properties

	Name	Description
	ClientCache	Gets the ClientCacheBase to which this client scope is connected.

Top

Public Methods

	Name	Description
	AddRef	Overloaded. Marks an entity as needed. Needed entities are not detached/released from the context until the client scope is disposed.
	Dispose	Marks the scope as disposed. Entities that were marked needed by a disposed scope may be disposed of (evicted from the cache, detached from context) unless they are needed by other active scopes.
	Release	Overloaded. Unmark a needed entity.

Top

See Also

Reference

ClientScope Class
C1.Data Namespace

ClientScope Constructor
An instance of the ClientCacheBase class to which the new client scopeis connected.

Initializes a new instance of ClientScope class with the given ClientCacheBase.

Syntax

Visual Basic (Declaration)

`Public Function New( _`
`  ByVal clientCache As ClientCacheBase _`
`)`

C#

`public ClientScope(`
`  ClientCacheBase clientCache`
`)`

Parameters

clientCache

An instance of the ClientCacheBase class to which the new client scopeis connected.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientScope Class
ClientScope Members

Methods

For a list of all members of this type, see ClientScope members.

Public Methods

	Name	Description
	AddRef	Overloaded. Marks an entity as needed. Needed entities are not detached/released from the context until the client scope is disposed.
	Dispose	Marks the scope as disposed. Entities that were marked needed by a disposed scope may be disposed of (evicted from the cache, detached from context) unless they are needed by other active scopes.
	Release	Overloaded. Unmark a needed entity.

Top

See Also

Reference

ClientScope Class
C1.Data Namespace

AddRef Method

Marks an entity as needed. Needed entities are not detached/released from the context until the client scope is disposed.

Overload List

Overload	Description
AddRef(Object)	Marks an entity as needed. Needed entities are not detached/released from the context until the client scope is disposed.
AddRef(Type)	Mark all entities of a given type as needed. All entities of that type will not be detached from the context until the client scope is disposed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientScope Class
ClientScope Members

AddRef(Object) Method
An entity to be marked as needed.

Marks an entity as needed. Needed entities are not detached/released from the context until the client scope is disposed.

Syntax

Visual Basic (Declaration)

```
Public Overloads Sub AddRef( _
   ByVal entity As Object _
)
```

C#

```
public void AddRef(
   object entity
)
```

Parameters

entity

An entity to be marked as needed.

Remarks

Client views and C1DataSource classes mark entities as needed automatically. Use this method only when you fetch entities using other means, bypassing Studio for EF classes with direct access to the underlying object context. When you no longer need those entities, call Release(Object). AddRef and Release are counting, every AddRef call must be balanced by a Release call.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientScope Class
ClientScope Members
Overload List

AddRef(Type) Method
An entity type to mark as needed.

Mark all entities of a given type as needed. All entities of that type will not be detached from the context until the client scope is disposed.

Syntax

Visual Basic (Declaration)

`Public Overloads Sub AddRef( _`
`  ByVal entityType As Type _`
`)`

C#

`public void AddRef(`
`  Type entityType`
`)`

Parameters

entityType

An entity type to mark as needed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientScope Class
ClientScope Members
Overload List

Dispose Method
Marks the scope as disposed. Entities that were marked needed by a disposed scope may be disposed of (evicted from the cache, detached from context) unless they are needed by other active scopes.

Syntax

Visual Basic (Declaration)
`Public Sub Dispose()`
C#
`public void Dispose()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientScope Class
ClientScope Members

Release Method
Unmark a needed entity.

Overload List

Overload	Description

Release(Object)	Unmark a needed entity.
Release(Type)	Unmark a needed entity type. Calling this method does not release memory.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientScope Class
ClientScope Members

Release(Object) Method
An entity that was marked as needed using AddRef(Object).

Unmark a needed entity.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function Release( _
   ByVal entity As Object _
) As Boolean
```

C#

```
public bool Release(
   object entity
)
```

Parameters

entity

An entity that was marked as needed using AddRef(Object).

Return Value

True if the entity was unmarked; otherwise, False (the entity is not unmarked until every AddRef is balanced by a Release).

Remarks

Calling this method does not release memory by itself. The entity becomes unneeded, so it can be disposed of at cache cleanup time.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientScope Class
ClientScope Members
Overload List

Release(Type) Method
An entity type that was marked as needed using AddRef(Type).

Unmark a needed entity type. Calling this method does not release memory.

Syntax

Visual Basic (Declaration)

`Public Overloads Function Release( _`
`  ByVal entityType As Type _`
`) As Boolean`

C#

`public bool Release(`
`  Type entityType`
`)`

Parameters

entityType

An entity type that was marked as needed using AddRef(Type).

Return Value

True if the entity type was unmarked (every Addref was balanced with Release); otherwise, False.

Remarks

Calling this method does not release memory until cache cleanup time.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientScope Class
ClientScope Members
Overload List

Properties
For a list of all members of this type, see ClientScope members.

Public Properties

	Name	Description
	ClientCache	Gets the ClientCacheBase to which this client scope is connected.

Top

See Also

Reference

ClientScope Class
C1.Data Namespace

ClientCache Property

Gets the ClientCacheBase to which this client scope is connected.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property ClientCache As ClientCacheBase`
C#
`public ClientCacheBase ClientCache {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientScope Class
ClientScope Members

ClientView<T>

The type of the elements in the view.

Represents a *client view*, that is, a live view that is connected to a remote source, such as an System.Data.Objects.ObjectContext.

Object Model

Syntax

Visual Basic (Declaration)

`Public Class ClientView(Of T)`
`  Inherits C1.LiveLinq.LiveViews.View(Of T)`
`  Implements C1.LiveLinq.Indexing.IIndexedSource(Of T),`

C1.LiveLinq.IObservableSource(Of T)	
C#	
`public class ClientView<T> : C1.LiveLinq.LiveViews.View<T>, C1.LiveLinq.Indexing.IIndexedSource<T>, C1.LiveLinq.IObservableSource<T>`	

Type Parameters

T

The type of the elements in the view.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.View
    C1.LiveLinq.LiveViews.View<T>
      C1.Data.ClientView<T>
        C1.Data.FilteredView<T>
        C1.Data.PagingView<T>
        C1.Data.ProgressiveView<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientView<T> Members
C1.Data Namespace

Overview
The type of the elements in the view.

Represents a *client view*, that is, a live view that is connected to a remote source, such as an System.Data.Objects.ObjectContext.

Object Model

ClientView<T>

Syntax

Visual Basic (Declaration)

`Public Class ClientView(Of T)`
`  Inherits C1.LiveLinq.LiveViews.View(Of T)`
`  Implements C1.LiveLinq.Indexing.IIndexedSource(Of T),`
`C1.LiveLinq.IObservableSource(Of T)`

C#
`public class ClientView<T> : C1.LiveLinq.LiveViews.View<T>,` `C1.LiveLinq.Indexing.IIndexedSource<T>, C1.LiveLinq.IObservableSource<T>`

Type Parameters

T

The type of the elements in the view.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.View
    C1.LiveLinq.LiveViews.View<T>
      C1.Data.ClientView<T>
        C1.Data.FilteredView<T>
        C1.Data.PagingView<T>
        C1.Data.ProgressiveView<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientView<T> Members
C1.Data Namespace

Members

Properties Methods Events

The following tables list the members exposed by ClientView<T>.

Public Properties

	Name	Description
	AutoLoad	Gets or sets a boolean value indicating whether the client view must be loaded automatically when its data is accessed.
	Count	Gets the total number of elements in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	CurrentItem	Gets the current item in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	DataBindingMode	Gets or sets the data binding mode for this view. (Inherited from C1.LiveLinq.LiveViews.View)
	DeferredMaintenance	Gets the effective value of MaintenanceMode. (Inherited from C1.LiveLinq.LiveViews.View)
	Indexes	Gets the collection of indexes for this view. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	IsLoaded	Gets a value indicating whether the client view is loaded.
	IsLoading	Gets a value indicating whether the client view is being loaded.
	IsReadOnly	Gets a value indicating whether this view is read-only, not updatable. (Inherited from C1.LiveLinq.LiveViews.View)

	Item	Gets the view item (element) at the specified ordinal position. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	MaintenanceMode	Gets or sets a value controlling how the view is synchronized with changes in its base data. (Inherited from C1.LiveLinq.LiveViews.View)
	MoveToFirstOnReset	Gets or sets a value indicating that the first item must be made current after initial loading or reset (on any C1.LiveLinq.SourceChangeType.Reset notification) if current item was not set by other means. The default is True. (Inherited from C1.LiveLinq.LiveViews.View)
	Order	Gets a value indicating whether and how this view preserves item order if it exists in its base data source. (Inherited from C1.LiveLinq.LiveViews.View)
	Scope	Gets the client scope to which this client view belongs.
	Transaction	Gets an instance of C1.LiveLinq.ITransaction associated with the view. If a view has a transaction associated with it, that transaction's scope is opened automatically every time the view is updated, so the programmer does not need to do it manually in code. (Inherited from C1.LiveLinq.LiveViews.View)

Top

Public Methods

	Name	Description
	AsCollectionViewFactory	Returns an instance of System.ComponentModel.ICollectionViewFactory that can be used as a source of a System.Windows.Data.CollectionViewSource. (Inherited from C1.LiveLinq.LiveViews.View)

	AsDynamic	Used for views with anonymous type constructor as the result selector, converts the C1.LiveLinq.LiveViews.View to a View<dynamic> so it can be used for data binding and programmatic access. (Inherited from C1.LiveLinq.LiveViews.View)
	AsFiltered	Filters the view on the server side using a predicate.
	AsFilteredBound<TKey>	Filters the view on the server using a key selector function and configurable value and operator.
	AttachAggregationView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	AttachView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	CancelLoad	Cancels the current loading operation.
	Concat	Concatenation of two views. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Contains	Determines whether the view contains a specified item. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	DeferMaintenance	Enters a defer cycle that you can use to make bulk changes to the view sources and delay automatic view maintenance. (Inherited from C1.LiveLinq.LiveViews.View)
	GetEnumerator	Returns an enumerator that iterates through the view items. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	GroupBy	Overloaded. Groups the elements of a view according to a specified key selector function. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	GroupJoin<TInner,TKey,TResult>	Correlates the elements of two views based on equality of

		keys and groups the results. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Include	Specifies related objects to include while loading the client view.
	IndexOf	Searches for the specified object among the elements of the view and returns the zero-based ordinal position of its first occurrence. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Join<TInner,TKey,TResult>	Correlates the elements of two views based on matching keys. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Load	Loads the entities of the client view.
	Maintain	Brings the view up to date with its source data. (Inherited from C1.LiveLinq.LiveViews.View)
	OrderBy<TKey>	Sorts the elements of a view in ascending order. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	OrderByDescending<TKey>	Sorts the elements of a view in descending order. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Paging	Overloaded. Applies paging to this client view.
	ProgressiveLoading	Overloaded. Specifies that the client view loading is performed not in a single trip to the server but in multiple batches, each loading a page of a limited size so the user sees the result and can interact with it before all pages are loaded.
	PurgeEmptyGroups	Remove empty groups from a grouping view. (Inherited from C1.LiveLinq.LiveViews.View)

	Rebuild	Re-populates the view by re-executing the view's query. (Inherited from C1.LiveLinq.LiveViews.View)
	Refresh	Loads the entities of the client view ignoring the client-side cache.
	Select<TResult>	Projects each element of a view into a new form. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	SelectMany	Overloaded. Projects each element of this view to a collection of collections, flattens the resulting collections into one collection, and invokes a result selector function on each element therein. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	SetTransaction	Sets the value of the C1.LiveLinq.LiveViews.View.Transaction property. (Inherited from C1.LiveLinq.LiveViews.View)
	ToString	Returns a string representing this view. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Union	Set union of two views. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Where	Filters the source view based on a predicate. (Inherited from C1.LiveLinq.LiveViews.View<T>)

Top

Public Events

	Name	Description
	Changed	Occurs after an item of the view or the entire view has changed. (Inherited from C1.LiveLinq.LiveViews.View<T>)

	Loaded	Occurs when the client view has finished loading succesfully, and also when an exception has been thrown during loading.

Top

See Also

Reference

ClientView<T> Class
C1.Data Namespace

Methods
For a list of all members of this type, see ClientView<T> members.

Public Methods

	Name	Description
	AsCollectionViewFactory	Returns an instance of System.ComponentModel.ICollectionViewFactory that can be used as a source of a System.Windows.Data.CollectionViewSource. (Inherited from C1.LiveLinq.LiveViews.View)
	AsDynamic	Used for views with anonymous type constructor as the result selector, converts the C1.LiveLinq.LiveViews.View to a View<dynamic> so it can be used for data binding and programmatic access. (Inherited from C1.LiveLinq.LiveViews.View)
	AsFiltered	Filters the view on the server side using a predicate.
	AsFilteredBound<TKey>	Filters the view on the server using a key selector function and configurable value and operator.
	AttachAggregationView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<T>)

	AttachView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	CancelLoad	Cancels the current loading operation.
	Concat	Concatenation of two views. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Contains	Determines whether the view contains a specified item. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	DeferMaintenance	Enters a defer cycle that you can use to make bulk changes to the view sources and delay automatic view maintenance. (Inherited from C1.LiveLinq.LiveViews.View)
	GetEnumerator	Returns an enumerator that iterates through the view items. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	GroupBy	Overloaded. Groups the elements of a view according to a specified key selector function. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	GroupJoin<TInner,TKey,TResult>	Correlates the elements of two views based on equality of keys and groups the results. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Include	Specifies related objects to include while loading the client view.
	IndexOf	Searches for the specified object among the elements of the view and returns the zero-based ordinal position of its first occurrence. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Join<TInner,TKey,TResult>	Correlates the elements of two views based on matching keys. (Inherited from C1.LiveLinq.LiveViews.View<T>)

	Load	Loads the entities of the client view.
	Maintain	Brings the view up to date with its source data. (Inherited from C1.LiveLinq.LiveViews.View)
	OrderBy<TKey>	Sorts the elements of a view in ascending order. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	OrderByDescending<TKey>	Sorts the elements of a view in descending order. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Paging	Overloaded. Applies paging to this client view.
	ProgressiveLoading	Overloaded. Specifies that the client view loading is performed not in a single trip to the server but in multiple batches, each loading a page of a limited size so the user sees the result and can interact with it before all pages are loaded.
	PurgeEmptyGroups	Remove empty groups from a grouping view. (Inherited from C1.LiveLinq.LiveViews.View)
	Rebuild	Re-populates the view by re-executing the view's query. (Inherited from C1.LiveLinq.LiveViews.View)
	Refresh	Loads the entities of the client view ignoring the client-side cache.
	Select<TResult>	Projects each element of a view into a new form. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	SelectMany	Overloaded. Projects each element of this view to a collection of collections, flattens the resulting collections into one collection, and invokes a result selector function on each element therein. (Inherited from

		C1.LiveLinq.LiveViews.View<T>)
	SetTransaction	Sets the value of the C1.LiveLinq.LiveViews.View.Transaction property. (Inherited from C1.LiveLinq.LiveViews.View)
	ToString	Returns a string representing this view. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Union	Set union of two views. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Where	Filters the source view based on a predicate. (Inherited from C1.LiveLinq.LiveViews.View<T>)

Top

See Also

Reference

ClientView<T> Class
C1.Data Namespace

AsFiltered Method
A function to apply each element to test the condition.

Filters the view on the server side using a predicate.

Syntax

Visual Basic (Declaration)

```
Public Overridable Function AsFiltered( _
   ByVal predicate As Expression(Of Func(Of T,Boolean)) _
) As ClientView(Of T)
```

C#

```
public virtual ClientView<T> AsFiltered(
   Expression<Func<T,bool>> predicate
```

```
)
```

Parameters

predicate

A function to apply each element to test the condition.

Return Value

A client view that contains elements of this view that satisfy the predicate.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientView<T> Class
ClientView<T> Members

AsFilteredBound<TKey> Method
The type of the value used for filtering.

A function that returns a key value for filtering given a view item.

Filters the view on the server using a key selector function and configurable value and operator.

Syntax

Visual Basic (Declaration)

```
Public Overridable Function AsFilteredBound(Of TKey)( _
   ByVal keySelector As Expression(Of Func(Of T,TKey)) _
) As FilteredView(Of T)
```

C#

```
public virtual FilteredView<T> AsFilteredBound<TKey>(
   Expression<Func<T,TKey>> keySelector
)
```

Parameters

keySelector

A function that returns a key value for filtering given a view item.

Type Parameters

TKey

The type of the value used for filtering.

Return Value

A FilteredView<T> that contains elements of this view that have keys satisfying the condition.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientView<T> Class
ClientView<T> Members

CancelLoad Method

Cancels the current loading operation.

Syntax

Visual Basic (Declaration)
`Public Sub CancelLoad()`
C#
`public void CancelLoad()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientView<T> Class
ClientView<T> Members

Include Method
A dot-separated list of related objects to load along with the entities of this client view.

Specifies related objects to include while loading the client view.

Syntax

Visual Basic (Declaration)

```
Public Overridable Function Include( _
   ByVal path As String _
) As ClientView(Of T)
```

C#

```
public virtual ClientView<T> Include(
   string path
)
```

Parameters

path

A dot-separated list of related objects to load along with the entities of this client view.

Return Value

A client view that loads the related objects together with its entities every time it fetches entities from the server.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientView<T> Class
ClientView<T> Members

Load Method
Loads the entities of the client view.

Syntax

Visual Basic (Declaration)
`Public Sub Load()`
C#
`public void Load()`

Remarks

If the entities are already in the cache, there will be no roundtrip to the server.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientView<T> Class
ClientView<T> Members

Paging Method
Applies paging to this client view.

Overload List

Overload	Description
Paging<TKey>(Expression<Func<T,TKey>>,Int32)	Applies paging to this client view.
Paging<TKey>(Expression<Func<T,TKey>>,Boolean,Int32)	Applies paging to this client view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientView<T> Class
ClientView<T> Members

Paging<TKey>(Expression<Func<T,TKey>>,Int32) Method
The type of the sort key.

A function specifying a sort key.

A value for the PageSize property, the number of items to load in a page.

Applies paging to this client view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function Paging(Of TKey)( _
   ByVal sortKeySelector As Expression(Of Func(Of T,TKey)), _
   ByVal pageSize As Integer _
) As PagingView(Of T)
```

C#

```
public PagingView<T> Paging<TKey>(
   Expression<Func<T,TKey>> sortKeySelector,
   int pageSize
```

```
)
```

Parameters

sortKeySelector

A function specifying a sort key.

pageSize

A value for the PageSize property, the number of items to load in a page.

Type Parameters

TKey

The type of the sort key.

Return Value

A paged client view.

Remarks

Sorting is required, paging entities is impossible without sort.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientView<T> Class
ClientView<T> Members
Overload List

Paging<TKey>(Expression<Func<T,TKey>>,Boolean,Int32) Method
The type of the sort key.

A function specifying a sort key.

A boolean value indicating whether sorting must be performed in the ascending order (descending, if false).

A value for the PageSize property, the number of items to load in a page.

Applies paging to this client view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function Paging(Of TKey)( _
   ByVal sortKeySelector As Expression(Of Func(Of T,TKey)), _
   ByVal ascending As Boolean, _
   ByVal pageSize As Integer _
) As PagingView(Of T)
```

C#

```
public PagingView<T> Paging<TKey>(
   Expression<Func<T,TKey>> sortKeySelector,
   bool ascending,
   int pageSize
)
```

Parameters

sortKeySelector

A function specifying a sort key.

ascending

A boolean value indicating whether sorting must be performed in the ascending order (descending, if false).

pageSize

A value for the PageSize property, the number of items to load in a page.

Type Parameters

TKey

The type of the sort key.

Return Value

A paged client view.

Remarks

Sorting is required, paging entities is impossible without sort.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientView<T> Class
ClientView<T> Members
Overload List

ProgressiveLoading Method
Specifies that the client view loading is performed not in a single trip to the server but in multiple batches, each loading a page of a limited size so the user sees the result and can interact with it before all pages are loaded.

Overload List

Overload	Description
ProgressiveLoading<TKey>(Expression<Func<T,TKey>>,Int32)	Specifies that the client view loading is performed not in a single trip to the server but in multiple batches, each loading a page of a limited size so the user sees the result and can interact with it before all pages are loaded.
ProgressiveLoading<TKey>(Expression<Func<T,TKey>>,Boolean,Int32)	Specifies that the client view loading is performed not in a single trip to the

	server but in multiple batches, each loading a page of a limited size so the user sees the result and can interact with it before all pages are loaded.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientView<T> Class
ClientView<T> Members

ProgressiveLoading<TKey>(Expression<Func<T,TKey>>,Int32) Method
The type of the sort key.

A function specifying a sort key.

The size of the page.

Specifies that the client view loading is performed not in a single trip to the server but in multiple batches, each loading a page of a limited size so the user sees the result and can interact with it before all pages are loaded.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function ProgressiveLoading(Of TKey)( _
   ByVal sortKeySelector As Expression(Of Func(Of T,TKey)), _
   ByVal loadSize As Integer _
) As ProgressiveView(Of T)
```

C#

```
public ProgressiveView<T> ProgressiveLoading<TKey>(
   Expression<Func<T,TKey>> sortKeySelector,
   int loadSize
)
```

Parameters

sortKeySelector

A function specifying a sort key.

loadSize

The size of the page.

Type Parameters

TKey

The type of the sort key.

Return Value

A client view that loads the same entities as the source view but does it progressively.

Remarks

Sorting is required, loading entities progressively is impossible without sort.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientView<T> Class
ClientView<T> Members
Overload List

ProgressiveLoading<TKey>(Expression<Func<T,TKey>>,Boolean,Int32) Method
The type of the sort key.

A function specifying a sort key.

A boolean value indicating whether sorting must be performed in the ascending order (descending, if false).

The size of the page.

Specifies that the client view loading is performed not in a single trip to the server but in multiple batches, each loading a page of a limited size so the user sees the result and can interact with it before all pages are loaded.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function ProgressiveLoading(Of TKey)( _
   ByVal sortKeySelector As Expression(Of Func(Of T,TKey)), _
   ByVal ascending As Boolean, _
   ByVal loadSize As Integer _
) As ProgressiveView(Of T)
```

C#

```
public ProgressiveView<T> ProgressiveLoading<TKey>(
   Expression<Func<T,TKey>> sortKeySelector,
   bool ascending,
   int loadSize
)
```

Parameters

sortKeySelector

A function specifying a sort key.

ascending

A boolean value indicating whether sorting must be performed in the ascending order (descending, if false).

loadSize

The size of the page.

Type Parameters

TKey

The type of the sort key.

Return Value

A client view that loads the same entities as the source view but does it progressively.

Remarks

Sorting is required, loading entities progressively is impossible without sort.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientView<T> Class
ClientView<T> Members
Overload List

Refresh Method

Loads the entities of the client view ignoring the client-side cache.

Syntax

Visual Basic (Declaration)
`Public Sub Refresh()`
C#
`public void Refresh()`

Remarks

Use this method to refresh data with any changes that may have occurred on the server

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientView<T> Class
ClientView<T> Members

Properties
For a list of all members of this type, see ClientView<T> members.

Public Properties

	Name	Description
	AutoLoad	Gets or sets a boolean value indicating whether the client view must be loaded automatically when its data is accessed.
	Count	Gets the total number of elements in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	CurrentItem	Gets the current item in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	DataBindingMode	Gets or sets the data binding mode for this view. (Inherited from C1.LiveLinq.LiveViews.View)
	DeferredMaintenance	Gets the effective value of MaintenanceMode. (Inherited from C1.LiveLinq.LiveViews.View)
	Indexes	Gets the collection of indexes for this view. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	IsLoaded	Gets a value indicating whether the client view is loaded.

	IsLoading	Gets a value indicating whether the client view is being loaded.
	IsReadOnly	Gets a value indicating whether this view is read-only, not updatable. (Inherited from C1.LiveLinq.LiveViews.View)
	Item	Gets the view item (element) at the specified ordinal position. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	MaintenanceMode	Gets or sets a value controlling how the view is synchronized with changes in its base data. (Inherited from C1.LiveLinq.LiveViews.View)
	MoveToFirstOnReset	Gets or sets a value indicating that the first item must be made current after initial loading or reset (on any C1.LiveLinq.SourceChangeType.Reset notification) if current item was not set by other means. The default is True. (Inherited from C1.LiveLinq.LiveViews.View)
	Order	Gets a value indicating whether and how this view preserves item order if it exists in its base data source. (Inherited from C1.LiveLinq.LiveViews.View)
	Scope	Gets the client scope to which this client view belongs.
	Transaction	Gets an instance of C1.LiveLinq.ITransaction associated with the view. If a view has a transaction associated with it, that transaction's scope is opened automatically every time the view is updated, so the programmer does not need to do it manually in code. (Inherited from C1.LiveLinq.LiveViews.View)

Top

See Also

Reference

ClientView<T> Class
C1.Data Namespace

AutoLoad Property

Gets or sets a boolean value indicating whether the client view must be loaded automatically when its data is accessed.

Syntax

Visual Basic (Declaration)
`Public Property AutoLoad As Boolean`
C#
`public bool AutoLoad {get; set;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientView<T> Class
ClientView<T> Members

IsLoaded Property

Gets a value indicating whether the client view is loaded.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property IsLoaded As Boolean`
C#
`public bool IsLoaded {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientView<T> Class
ClientView<T> Members

IsLoading Property
Gets a value indicating whether the client view is being loaded.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property IsLoading As Boolean`
C#
`public bool IsLoading {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientView<T> Class
ClientView<T> Members

Scope Property
Gets the client scope to which this client view belongs.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Scope As ClientScope`
C#

```
public ClientScope Scope {get;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientView<T> Class
ClientView<T> Members

Events
>

Name	Description
Changed	Occurs after an item of the view or the entire view has changed. (Inherited from C1.LiveLinq.LiveViews.View<T>)
Loaded	Occurs when the client view has finished loading succesfully, and also when an exception has been thrown during loading.

Top

See Also

Reference

ClientView<T> Class
C1.Data Namespace

Loaded Event
Occurs when the client view has finished loading succesfully, and also when an exception has been thrown during loading.

Syntax

Visual Basic (Declaration)	
`Public Event Loaded As EventHandler(Of ClientViewLoadedEventArgs)`	

C#	
`public event EventHandler<ClientViewLoadedEventArgs> Loaded`	

Event Data

The event handler receives an argument of type ClientViewLoadedEventArgs containing data related to this event. The following **ClientViewLoadedEventArgs** properties provide information specific to this event.

Property	Description
Error	Gets the loading error if the loading failed.
HasError	Gets a value indicating whether the loading has failed. If true, inspect the Error property for details.
IsErrorHandled	Gets a value indicating whether the loading error has been marked as handled by calling MarkErrorAsHandled.
Items	Gets all entities loaded by a client view.
TotalItemCount	Gets the total number of rows for the original query without any paging applied to it.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientView<T> Class
ClientView<T> Members

ClientViewLoadedEventArgs
Provides data for the ClientView<T>.Loaded event.

Object Model

ClientViewLoadedEventArgs

Syntax

Visual Basic (Declaration)
`Public Class ClientViewLoadedEventArgs` `  Inherits System.EventArgs`
C#
`public class ClientViewLoadedEventArgs : System.EventArgs`

Inheritance Hierarchy

System.Object
  System.EventArgs
    C1.Data.ClientViewLoadedEventArgs

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewLoadedEventArgs Members
C1.Data Namespace

Overview
Provides data for the ClientView<T>.Loaded event.

Object Model

ClientViewLoadedEventArgs

Syntax

Visual Basic (Declaration)
`Public Class ClientViewLoadedEventArgs` `  Inherits System.EventArgs`
C#
`public class ClientViewLoadedEventArgs : System.EventArgs`

Inheritance Hierarchy

System.Object
  System.EventArgs
    C1.Data.ClientViewLoadedEventArgs

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewLoadedEventArgs Members
C1.Data Namespace

Members
Properties Methods

The following tables list the members exposed by ClientViewLoadedEventArgs.

Public Properties

	Name	Description
	Error	Gets the loading error if the loading failed.
	HasError	Gets a value indicating whether the loading has failed. If true, inspect the Error property for details.

	IsErrorHandled	Gets a value indicating whether the loading error has been marked as handled by calling MarkErrorAsHandled.
	Items	Gets all entities loaded by a client view.
	TotalItemCount	Gets the total number of rows for the original query without any paging applied to it.

Top

Public Methods

	Name	Description
	MarkErrorAsHandled	For the case where HasError is true, this method marks the error as handled. If this method is not called, an exception will be thrown.

Top

See Also

Reference

ClientViewLoadedEventArgs Class
C1.Data Namespace

Methods
For a list of all members of this type, see ClientViewLoadedEventArgs members.

Public Methods

	Name	Description
	MarkErrorAsHandled	For the case where HasError is true, this method marks the error as handled. If this method is not called, an exception will be thrown.

Top

See Also

Reference

ClientViewLoadedEventArgs Class
C1.Data Namespace

MarkErrorAsHandled Method

For the case where HasError is true, this method marks the error as handled. If this method is not called, an exception will be thrown.

Syntax

Visual Basic (Declaration)
`Public Sub MarkErrorAsHandled()`
C#
`public void MarkErrorAsHandled()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewLoadedEventArgs Class
ClientViewLoadedEventArgs Members

Properties

For a list of all members of this type, see ClientViewLoadedEventArgs members.

Public Properties

	Name	Description
	Error	Gets the loading error if the loading failed.
	HasError	Gets a value indicating whether the loading has failed. If true, inspect the Error property for details.

	IsErrorHandled	Gets a value indicating whether the loading error has been marked as handled by calling MarkErrorAsHandled.
	Items	Gets all entities loaded by a client view.
	TotalItemCount	Gets the total number of rows for the original query without any paging applied to it.

Top

See Also

Reference

ClientViewLoadedEventArgs Class
C1.Data Namespace

Error Property
Gets the loading error if the loading failed.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Error As Exception`
C#
`public Exception Error {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewLoadedEventArgs Class
ClientViewLoadedEventArgs Members

HasError Property

Gets a value indicating whether the loading has failed. If true, inspect the Error property for details.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property HasError As Boolean`
C#
`public bool HasError {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewLoadedEventArgs Class
ClientViewLoadedEventArgs Members

IsErrorHandled Property

Gets a value indicating whether the loading error has been marked as handled by calling MarkErrorAsHandled.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property IsErrorHandled As Boolean`
C#
`public bool IsErrorHandled {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewLoadedEventArgs Class
ClientViewLoadedEventArgs Members

Items Property
Gets all entities loaded by a client view.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Items As IEnumerable`
C#
`public IEnumerable Items {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewLoadedEventArgs Class
ClientViewLoadedEventArgs Members

TotalItemCount Property
Gets the total number of rows for the original query without any paging applied to it.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property TotalItemCount As Integer`
C#

```
public int TotalItemCount {get;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

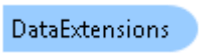
See Also

Reference

ClientViewLoadedEventArgs Class
ClientViewLoadedEventArgs Members

DataExtensions
Extension methods provided by Studio for Entity Framework.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit NotInheritable Class DataExtensions`
C#
`public static class DataExtensions`

Inheritance Hierarchy

System.Object
  C1.Data.DataExtensions

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

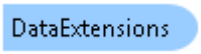
See Also

Reference

DataExtensions Members
C1.Data Namespace

Overview
Extension methods provided by Studio for Entity Framework.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit NotInheritable Class DataExtensions`
C#
`public static class DataExtensions`

Inheritance Hierarchy

System.Object
  C1.Data.DataExtensions

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

DataExtensions Members
C1.Data Namespace

Members
Methods

The following tables list the members exposed by DataExtensions.

Public Methods

	Name	Description
S	ExecuteIn<T>	Executes a query in a scope, so the loaded entities are pinned to the scope (marked as needed) by calling ClientScope.AddRef for each of them.

Top

See Also

Reference

DataExtensions Class
C1.Data Namespace

Methods
For a list of all members of this type, see DataExtensions members.

Public Methods

	Name	Description
S	ExecuteIn<T>	Executes a query in a scope, so the loaded entities are pinned to the scope (marked as needed) by calling ClientScope.AddRef for each of them.

Top

See Also

Reference

DataExtensions Class
C1.Data Namespace

ExecuteIn<T> Method
The type of items returned by the query.

The query to execute inside the client scope.

The client scope to execute the query in.

Executes a query in a scope, so the loaded entities are pinned to the scope (marked as needed) by calling ClientScope.AddRef for each of them.

Syntax

Visual Basic (Declaration)

```
Public Shared Function ExecuteIn(Of T)( _
   ByVal query As IEnumerable(Of T), _
   ByVal scope As ClientScope _
) As IEnumerable(Of T)
```

C#

```
public static IEnumerable<T> ExecuteIn<T>(
   IEnumerable<T> query,
   ClientScope scope
)
```

Parameters

query

The query to execute inside the client scope.

scope

The client scope to execute the query in.

Type Parameters

T

The type of items returned by the query.

Return Value

The query that will be executed inside the given scope.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

DataExtensions Class
DataExtensions Members

FilteredView<T>
The type of the entities in this client view.

Represents a client view filtered by a filter key function, an operator and a filter key value.

Object Model

Syntax

Visual Basic (Declaration)

`Public Class FilteredView(Of T)`
`  Inherits ClientView(Of T)`
`  Implements C1.LiveLinq.Indexing.IIndexedSource(Of T),`
`C1.LiveLinq.IObservableSource(Of T)`

C#
`public class FilteredView<T> : ClientView<T>,` `C1.LiveLinq.Indexing.IIndexedSource<T>, C1.LiveLinq.IObservableSource<T>`

Type Parameters

T

The type of the entities in this client view.

Remarks

Filtering is performed on the server.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.View
    C1.LiveLinq.LiveViews.View<T>

C1.Data.ClientView<T>
C1.Data.FilteredView<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

FilteredView<T> Members
C1.Data Namespace

Overview
The type of the entities in this client view.

Represents a client view filtered by a filter key function, an operator and a filter key value.

Object Model

Syntax

Visual Basic (Declaration)	

`Public Class FilteredView(Of T)`
`  Inherits ClientView(Of T)`
`  Implements C1.LiveLinq.Indexing.IIndexedSource(Of T),`
`C1.LiveLinq.IObservableSource(Of T)`	

C#	
`public class FilteredView<T> : ClientView<T>,` `C1.LiveLinq.Indexing.IIndexedSource<T>, C1.LiveLinq.IObservableSource<T>`	

Type Parameters

T

The type of the entities in this client view.

Remarks

Filtering is performed on the server.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.View
    C1.LiveLinq.LiveViews.View<T>
      C1.Data.ClientView<T>
        C1.Data.FilteredView<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

FilteredView<T> Members
C1.Data Namespace

Members
Fields Properties Methods Events

The following tables list the members exposed by FilteredView<T>.

Public Fields

	Name	Description
S	Unfiltered	A special value indicating that filtering must not be performed. To disable filtering, assign the value of this field to the FilterKey property.

Top

Public Properties

	Name	Description

	AutoLoad	Gets or sets a boolean value indicating whether the client view must be loaded automatically when its data is accessed. (Inherited from C1.Data.ClientView<T>)
	Count	Gets the total number of elements in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	CurrentItem	Gets the current item in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	DataBindingMode	Gets or sets the data binding mode for this view. (Inherited from C1.LiveLinq.LiveViews.View)
	DeferredMaintenance	Gets the effective value of MaintenanceMode. (Inherited from C1.LiveLinq.LiveViews.View)
	FilterKey	Gets or sets a value that is used to filter items by comparing this value to the result of the filter key function applied to an item.
	FilterKeyType	Gets the filter key type. It is determined by the filter key function.
	Indexes	Gets the collection of indexes for this view. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	IsLoaded	Gets a value indicating whether the client view is loaded. (Inherited from C1.Data.ClientView<T>)
	IsLoading	Gets a value indicating whether the client view is being loaded. (Inherited from C1.Data.ClientView<T>)
	IsReadOnly	Gets a value indicating whether this view is read-only, not updatable. (Inherited from C1.LiveLinq.LiveViews.View)
	Item	Gets the view item (element) at the specified ordinal position. (Inherited from C1.LiveLinq.LiveViews.View<T>)

	MaintenanceMode	Gets or sets a value controlling how the view is synchronized with changes in its base data. (Inherited from C1.LiveLinq.LiveViews.View)
	MoveToFirstOnReset	Gets or sets a value indicating that the first item must be made current after initial loading or reset (on any C1.LiveLinq.SourceChangeType.Reset notification) if current item was not set by other means. The default is True. (Inherited from C1.LiveLinq.LiveViews.View)
	Operator	Gets a C1.Data.DataSource.FilterOperator used to compare filter keys.
	Order	Gets a value indicating whether and how this view preserves item order if it exists in its base data source. (Inherited from C1.LiveLinq.LiveViews.View)
	Scope	Gets the client scope to which this client view belongs. (Inherited from C1.Data.ClientView<T>)
	Transaction	Gets an instance of C1.LiveLinq.ITransaction associated with the view. If a view has a transaction associated with it, that transaction's scope is opened automatically every time the view is updated, so the programmer does not need to do it manually in code. (Inherited from C1.LiveLinq.LiveViews.View)

Top

Public Methods

	Name	Description
	AsCollectionViewFactory	Returns an instance of System.ComponentModel.ICollectionViewFactory that can be used as a source of a System.Windows.Data.CollectionViewSource. (Inherited from C1.LiveLinq.LiveViews.View)

	AsDynamic	Used for views with anonymous type constructor as the result selector, converts the C1.LiveLinq.LiveViews.View to a View<dynamic> so it can be used for data binding and programmatic access. (Inherited from C1.LiveLinq.LiveViews.View)
	AsFiltered	Filters the view on the server side using a predicate. (Inherited from C1.Data.ClientView<T>)
	AsFilteredBound<TKey>	Filters the view on the server using a key selector function and configurable value and operator. (Inherited from C1.Data.ClientView<T>)
	AttachAggregationView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	AttachView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	BindFilterKey	Overloaded. Binds the FilterKey property using the specified System.Windows.Data.Binding object.
	CancelLoad	Cancels the current loading operation. (Inherited from C1.Data.ClientView<T>)
	Concat	Concatenation of two views. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Contains	Determines whether the view contains a specified item. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	DeferMaintenance	Enters a defer cycle that you can use to make bulk changes to the view sources and delay automatic view maintenance. (Inherited from C1.LiveLinq.LiveViews.View)
	GetEnumerator	Returns an enumerator that iterates through the view items.

		(Inherited from C1.LiveLinq.LiveViews.View<T>)
	GroupBy	Overloaded. Groups the elements of a view according to a specified key selector function. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	GroupJoin<TInner,TKey,TResult>	Correlates the elements of two views based on equality of keys and groups the results. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Include	Specifies related objects to include while loading the client view. (Inherited from C1.Data.ClientView<T>)
	IndexOf	Searches for the specified object among the elements of the view and returns the zero-based ordinal position of its first occurrence. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Join<TInner,TKey,TResult>	Correlates the elements of two views based on matching keys. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Load	Loads the entities of the client view. (Inherited from C1.Data.ClientView<T>)
	Maintain	Brings the view up to date with its source data. (Inherited from C1.LiveLinq.LiveViews.View)
	OrderBy<TKey>	Sorts the elements of a view in ascending order. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	OrderByDescending<TKey>	Sorts the elements of a view in descending order. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Paging	Overloaded. Applies paging to this client view. (Inherited from C1.Data.ClientView<T>)

	ProgressiveLoading	Overloaded. Specifies that the client view loading is performed not in a single trip to the server but in multiple batches, each loading a page of a limited size so the user sees the result and can interact with it before all pages are loaded. (Inherited from C1.Data.ClientView<T>)
	PurgeEmptyGroups	Remove empty groups from a grouping view. (Inherited from C1.LiveLinq.LiveViews.View)
	Rebuild	Re-populates the view by re-executing the view's query. (Inherited from C1.LiveLinq.LiveViews.View)
	Refresh	Loads the entities of the client view ignoring the client-side cache. (Inherited from C1.Data.ClientView<T>)
	Select<TResult>	Projects each element of a view into a new form. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	SelectMany	Overloaded. Projects each element of this view to a collection of collections, flattens the resulting collections into one collection, and invokes a result selector function on each element therein. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	SetTransaction	Sets the value of the C1.LiveLinq.LiveViews.View.Transaction property. (Inherited from C1.LiveLinq.LiveViews.View)
	ToString	Returns a string representing this view. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Union	Set union of two views. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Where	Filters the source view based on a predicate. (Inherited from

		C1.LiveLinq.LiveViews.View<T>)

Top

Public Events

	Name	Description
	Changed	Occurs after an item of the view or the entire view has changed. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Loaded	Occurs when the client view has finished loading succesfully, and also when an exception has been thrown during loading. (Inherited from C1.Data.ClientView<T>)

Top

See Also

Reference

FilteredView<T> Class
C1.Data Namespace

Methods
For a list of all members of this type, see FilteredView<T> members.

Public Methods

	Name	Description
	AsCollectionViewFactory	Returns an instance of System.ComponentModel.ICollectionViewFactory that can be used as a source of a System.Windows.Data.CollectionViewSource. (Inherited from C1.LiveLinq.LiveViews.View)
	AsDynamic	Used for views with anonymous type constructor as the result selector, converts the C1.LiveLinq.LiveViews.View to a

		View<dynamic> so it can be used for data binding and programmatic access. (Inherited from C1.LiveLinq.LiveViews.View)
	AsFiltered	Filters the view on the server side using a predicate. (Inherited from C1.Data.ClientView<T>)
	AsFilteredBound<TKey>	Filters the view on the server using a key selector function and configurable value and operator. (Inherited from C1.Data.ClientView<T>)
	AttachAggregationView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	AttachView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	BindFilterKey	Overloaded. Binds the FilterKey property using the specified System.Windows.Data.Binding object.
	CancelLoad	Cancels the current loading operation. (Inherited from C1.Data.ClientView<T>)
	Concat	Concatenation of two views. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Contains	Determines whether the view contains a specified item. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	DeferMaintenance	Enters a defer cycle that you can use to make bulk changes to the view sources and delay automatic view maintenance. (Inherited from C1.LiveLinq.LiveViews.View)
	GetEnumerator	Returns an enumerator that iterates through the view items. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	GroupBy	Overloaded. Groups the elements of a view according to a

		specified key selector function. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	GroupJoin<TInner,TKey,TResult>	Correlates the elements of two views based on equality of keys and groups the results. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Include	Specifies related objects to include while loading the client view. (Inherited from C1.Data.ClientView<T>)
	IndexOf	Searches for the specified object among the elements of the view and returns the zero-based ordinal position of its first occurrence. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Join<TInner,TKey,TResult>	Correlates the elements of two views based on matching keys. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Load	Loads the entities of the client view. (Inherited from C1.Data.ClientView<T>)
	Maintain	Brings the view up to date with its source data. (Inherited from C1.LiveLinq.LiveViews.View)
	OrderBy<TKey>	Sorts the elements of a view in ascending order. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	OrderByDescending<TKey>	Sorts the elements of a view in descending order. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Paging	Overloaded. Applies paging to this client view. (Inherited from C1.Data.ClientView<T>)
	ProgressiveLoading	Overloaded. Specifies that the client view loading is performed not in a single trip to the server but in multiple batches, each loading a page of a limited size so the user

		sees the result and can interact with it before all pages are loaded. (Inherited from C1.Data.ClientView<T>)
	PurgeEmptyGroups	Remove empty groups from a grouping view. (Inherited from C1.LiveLinq.LiveViews.View)
	Rebuild	Re-populates the view by re-executing the view's query. (Inherited from C1.LiveLinq.LiveViews.View)
	Refresh	Loads the entities of the client view ignoring the client-side cache. (Inherited from C1.Data.ClientView<T>)
	Select<TResult>	Projects each element of a view into a new form. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	SelectMany	Overloaded. Projects each element of this view to a collection of collections, flattens the resulting collections into one collection, and invokes a result selector function on each element therein. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	SetTransaction	Sets the value of the C1.LiveLinq.LiveViews.View.Transaction property. (Inherited from C1.LiveLinq.LiveViews.View)
	ToString	Returns a string representing this view. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Union	Set union of two views. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Where	Filters the source view based on a predicate. (Inherited from C1.LiveLinq.LiveViews.View<T>)

Top

See Also

Reference

FilteredView<T> Class
C1.Data Namespace

BindFilterKey Method
Binds the FilterKey property using the specified System.Windows.Data.Binding object.

Overload List

Overload	Description
BindFilterKey(Binding)	Binds the FilterKey property using the specified System.Windows.Data.Binding object.
BindFilterKey(Object,String)	Binds the FilterKey property to a given path on a given source.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

FilteredView<T> Class
FilteredView<T> Members

BindFilterKey(Binding) Method
The System.Windows.Data.Binding object used to bind the FilterKey. Use null to unbind the previously bound FilterKey.

Binds the FilterKey property using the specified System.Windows.Data.Binding object.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function BindFilterKey( _
   ByVal binding As Binding _
```

```
) As FilteredView(Of T)
```

C#

```
public FilteredView<T> BindFilterKey(
   Binding binding
)
```

Parameters

binding

The System.Windows.Data.Binding object used to bind the FilterKey. Use null to unbind the previously bound FilterKey.

Return Value

The same FilteredView<T> on which this method was called.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

FilteredView<T> Class
FilteredView<T> Members
Overload List

BindFilterKey(Object,String) Method
The object to bind to. Cannot be null.

The property path to bind to.

Binds the FilterKey property to a given path on a given source.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function BindFilterKey( _
```

```
   ByVal source As Object, _
   ByVal path As String _
) As FilteredView(Of T)
```

C#

```
public FilteredView<T> BindFilterKey(
   object source,
   string path
)
```

Parameters

source

The object to bind to. Cannot be null.

path

The property path to bind to.

Return Value

The same FilteredView<T> on which this method was called.

Exceptions

Exception	Description
System.NullReferenceException	source is null.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

FilteredView<T> Class
FilteredView<T> Members
Overload List

Properties

For a list of all members of this type, see FilteredView<T> members.

Public Properties

	Name	Description
	AutoLoad	Gets or sets a boolean value indicating whether the client view must be loaded automatically when its data is accessed. (Inherited from C1.Data.ClientView<T>)
	Count	Gets the total number of elements in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	CurrentItem	Gets the current item in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	DataBindingMode	Gets or sets the data binding mode for this view. (Inherited from C1.LiveLinq.LiveViews.View)
	DeferredMaintenance	Gets the effective value of MaintenanceMode. (Inherited from C1.LiveLinq.LiveViews.View)
	FilterKey	Gets or sets a value that is used to filter items by comparing this value to the result of the filter key function applied to an item.
	FilterKeyType	Gets the filter key type. It is determined by the filter key function.
	Indexes	Gets the collection of indexes for this view. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	IsLoaded	Gets a value indicating whether the client view is loaded. (Inherited from C1.Data.ClientView<T>)
	IsLoading	Gets a value indicating whether the client view is being loaded. (Inherited from C1.Data.ClientView<T>)

	IsReadOnly	Gets a value indicating whether this view is read-only, not updatable. (Inherited from C1.LiveLinq.LiveViews.View)
	Item	Gets the view item (element) at the specified ordinal position. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	MaintenanceMode	Gets or sets a value controlling how the view is synchronized with changes in its base data. (Inherited from C1.LiveLinq.LiveViews.View)
	MoveToFirstOnReset	Gets or sets a value indicating that the first item must be made current after initial loading or reset (on any C1.LiveLinq.SourceChangeType.Reset notification) if current item was not set by other means. The default is True. (Inherited from C1.LiveLinq.LiveViews.View)
	Operator	Gets a C1.Data.DataSource.FilterOperator used to compare filter keys.
	Order	Gets a value indicating whether and how this view preserves item order if it exists in its base data source. (Inherited from C1.LiveLinq.LiveViews.View)
	Scope	Gets the client scope to which this client view belongs. (Inherited from C1.Data.ClientView<T>)
	Transaction	Gets an instance of C1.LiveLinq.ITransaction associated with the view. If a view has a transaction associated with it, that transaction's scope is opened automatically every time the view is updated, so the programmer does not need to do it manually in code. (Inherited from C1.LiveLinq.LiveViews.View)

Top

See Also

Reference

FilteredView<T> Class
C1.Data Namespace

FilterKey Property
Gets or sets a value that is used to filter items by comparing this value to the result of the filter key function applied to an item.

Syntax

Visual Basic (Declaration)
`Public Property FilterKey As Object`
C#
`public object FilterKey {get; set;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

FilteredView<T> Class
FilteredView<T> Members
Operator Property

FilterKeyType Property
Gets the filter key type. It is determined by the filter key function.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property FilterKeyType As Type`
C#
`public Type FilterKeyType {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

FilteredView<T> Class
FilteredView<T> Members

Operator Property
Gets a C1.Data.DataSource.FilterOperator used to compare filter keys.

Syntax

Visual Basic (Declaration)
`Public Property Operator As FilterOperator`
C#
`public FilterOperator Operator {get; set;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

FilteredView<T> Class
FilteredView<T> Members
FilterOperator Enumeration

Fields
>

Name	Description
S Unfiltered	A special value indicating that filtering must not be performed. To disable

filtering, assign the value of this field to the FilterKey property.

Top

See Also

Reference

FilteredView<T> Class
C1.Data Namespace

Unfiltered Field
A special value indicating that filtering must not be performed. To disable filtering, assign the value of this field to the FilterKey property.

Syntax

Visual Basic (Declaration)
`Public Shared ReadOnly Unfiltered As Object`
C#
`public static readonly object Unfiltered`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

FilteredView<T> Class
FilteredView<T> Members

PageChangingEventArgs
Provides data for the C1.Data.DataSource.ClientCollectionView.PageChanging event.

Object Model

Syntax

Visual Basic (Declaration)	
`Public NotInheritable Class PageChangingEventArgs` `  Inherits System.ComponentModel.CancelEventArgs`	
C#	
`public sealed class PageChangingEventArgs :` `System.ComponentModel.CancelEventArgs`	

Inheritance Hierarchy

System.Object
  System.EventArgs
    System.ComponentModel.CancelEventArgs
      C1.Data.PageChangingEventArgs

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

PageChangingEventArgs Members
C1.Data Namespace

Overview
Provides data for the C1.Data.DataSource.ClientCollectionView.PageChanging event.

Object Model

Syntax

Visual Basic (Declaration)	

```
Public NotInheritable Class PageChangingEventArgs
   Inherits System.ComponentModel.CancelEventArgs
```

C#

```
public sealed class PageChangingEventArgs :
System.ComponentModel.CancelEventArgs
```

Inheritance Hierarchy

System.Object
  System.EventArgs
    System.ComponentModel.CancelEventArgs
      C1.Data.PageChangingEventArgs

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

PageChangingEventArgs Members
C1.Data Namespace

Members
Properties

The following tables list the members exposed by PageChangingEventArgs.

Public Constructors

	Name	Description
	PageChangingEventArgs Constructor	Initializes a new instance of the PageChangingEventArgs class.

Top

Public Properties

	Name	Description
	Cancel	(Inherited from System.ComponentModel.CancelEventArgs)
	NewPageIndex	Gets the index of the requested page.

Top

See Also

Reference

PageChangingEventArgs Class
C1.Data Namespace

PageChangingEventArgs Constructor
The index of the requested page.

Initializes a new instance of the PageChangingEventArgs class.

Syntax

Visual Basic (Declaration)

```
Public Function New( _
   ByVal newPageIndex As Integer _
)
```

C#

```
public PageChangingEventArgs(
   int newPageIndex
)
```

Parameters

newPageIndex

The index of the requested page.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

PageChangingEventArgs Class
PageChangingEventArgs Members

Properties
>

Name	Description
Cancel	(Inherited from System.ComponentModel.CancelEventArgs)
NewPageIndex	Gets the index of the requested page.

Top

See Also

Reference

PageChangingEventArgs Class
C1.Data Namespace

NewPageIndex Property
Gets the index of the requested page.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property NewPageIndex As Integer`
C#
`public int NewPageIndex {get;}`

Property Value

The index of the requested page.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

PageChangingEventArgs Class
PageChangingEventArgs Members

PagingView<T>
The type of the entities in the client view.

Represents a paged client view.

Object Model

Syntax

Visual Basic (Declaration)

`Public Class PagingView(Of T)`
`  Inherits ClientView(Of T)`
`  Implements C1.LiveLinq.Indexing.IIndexedSource(Of T),`
`C1.LiveLinq.IObservableSource(Of T)`

C#
`public class PagingView<T> : ClientView<T>,` `C1.LiveLinq.Indexing.IIndexedSource<T>, C1.LiveLinq.IObservableSource<T>`

Type Parameters

T

The type of the entities in the client view.

Remarks

Paging is performed on the server. It is controled by the PageSize and PageIndex properties.

Sorting is required, paging entities is impossible without sort.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.View
    C1.LiveLinq.LiveViews.View<T>
      C1.Data.ClientView<T>
        C1.Data.PagingView<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

PagingView<T> Members
C1.Data Namespace

Overview
The type of the entities in the client view.

Represents a paged client view.

Object Model

Syntax

Visual Basic (Declaration)	

```
Public Class PagingView(Of T)
   Inherits ClientView(Of T)
   Implements C1.LiveLinq.Indexing.IIndexedSource(Of T),
C1.LiveLinq.IObservableSource(Of T)
```

C#

```
public class PagingView<T> : ClientView<T>,
C1.LiveLinq.Indexing.IIndexedSource<T>, C1.LiveLinq.IObservableSource<T>
```

Type Parameters

T

The type of the entities in the client view.

Remarks

Paging is performed on the server. It is controled by the PageSize and PageIndex properties.

Sorting is required, paging entities is impossible without sort.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.View
    C1.LiveLinq.LiveViews.View<T>
      C1.Data.ClientView<T>
        C1.Data.PagingView<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

PagingView<T> Members
C1.Data Namespace

Members
Properties Methods Events

The following tables list the members exposed by PagingView<T>.

Public Properties

	Name	Description
	AutoLoad	Gets or sets a boolean value indicating whether the client view must be loaded automatically when its data is accessed. (Inherited from C1.Data.ClientView<T>)
	Count	Gets the total number of elements in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	CurrentItem	Gets the current item in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	DataBindingMode	Gets or sets the data binding mode for this view. (Inherited from C1.LiveLinq.LiveViews.View)
	DeferredMaintenance	Gets the effective value of MaintenanceMode. (Inherited from C1.LiveLinq.LiveViews.View)
	Indexes	Gets the collection of indexes for this view. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	IsLoaded	Gets a value indicating whether the client view is loaded. (Inherited from C1.Data.ClientView<T>)
	IsLoading	Gets a value indicating whether the client view is being loaded. (Inherited from C1.Data.ClientView<T>)
	IsReadOnly	Gets a value indicating whether this view is read-only, not updatable. (Inherited from C1.LiveLinq.LiveViews.View)
	Item	Gets the view item (element) at the specified ordinal position. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	LoadSize	Gets or sets a value controling the number of entities to load in one batch.

	MaintenanceMode	Gets or sets a value controlling how the view is synchronized with changes in its base data. (Inherited from C1.LiveLinq.LiveViews.View)
	MoveToFirstOnReset	Gets or sets a value indicating that the first item must be made current after initial loading or reset (on any C1.LiveLinq.SourceChangeType.Reset notification) if current item was not set by other means. The default is True. (Inherited from C1.LiveLinq.LiveViews.View)
	Order	Gets a value indicating whether and how this view preserves item order if it exists in its base data source. (Inherited from C1.LiveLinq.LiveViews.View)
	PageCount	Gets the number of pages in this view.
	PageIndex	Gets or sets the index of the current page.
	PageSize	Gets or sets the number of items in a page.
	Scope	Gets the client scope to which this client view belongs. (Inherited from C1.Data.ClientView<T>)
	TotalItemCount	Gets the total number of entities in the view before paging is applied.
	Transaction	Gets an instance of C1.LiveLinq.ITransaction associated with the view. If a view has a transaction associated with it, that transaction's scope is opened automatically every time the view is updated, so the programmer does not need to do it manually in code. (Inherited from C1.LiveLinq.LiveViews.View)

Top

Public Methods

	Name	Description

	AsCollectionViewFactory	Returns an instance of System.ComponentModel.ICollectionViewFactory that can be used as a source of a System.Windows.Data.CollectionViewSource. (Inherited from C1.LiveLinq.LiveViews.View)
	AsDynamic	Used for views with anonymous type constructor as the result selector, converts the C1.LiveLinq.LiveViews.View to a View<dynamic> so it can be used for data binding and programmatic access. (Inherited from C1.LiveLinq.LiveViews.View)
	AsFiltered	Filters the view on the server side using a predicate. (Inherited from C1.Data.ClientView<T>)
	AsFilteredBound<TKey>	Filters the view on the server using a key selector function and configurable value and operator. (Inherited from C1.Data.ClientView<T>)
	AttachAggregationView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	AttachView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	CancelLoad	Cancels the current loading operation. (Inherited from C1.Data.ClientView<T>)
	Concat	Concatenation of two views. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Contains	Determines whether the view contains a specified item. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	DeferMaintenance	Enters a defer cycle that you can use to make bulk changes to the view sources and delay automatic view maintenance.

		(Inherited from C1.LiveLinq.LiveViews.View)
	GetEnumerator	Returns an enumerator that iterates through the view items. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	GroupBy	Overloaded. Groups the elements of a view according to a specified key selector function. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	GroupJoin<TInner,TKey,TResult>	Correlates the elements of two views based on equality of keys and groups the results. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Include	Specifies related objects to include while loading the client view. (Inherited from C1.Data.ClientView<T>)
	IndexOf	Searches for the specified object among the elements of the view and returns the zero-based ordinal position of its first occurrence. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Join<TInner,TKey,TResult>	Correlates the elements of two views based on matching keys. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Load	Loads the entities of the client view. (Inherited from C1.Data.ClientView<T>)
	Maintain	Brings the view up to date with its source data. (Inherited from C1.LiveLinq.LiveViews.View)
	OrderBy<TKey>	Sorts the elements of a view in ascending order. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	OrderByDescending<TKey>	Sorts the elements of a view in descending order. (Inherited from C1.LiveLinq.LiveViews.View<T>)

	Paging	Overloaded. Applies paging to this client view. (Inherited from C1.Data.ClientView<T>)
	ProgressiveLoading	Overloaded. Specifies that the client view loading is performed not in a single trip to the server but in multiple batches, each loading a page of a limited size so the user sees the result and can interact with it before all pages are loaded. (Inherited from C1.Data.ClientView<T>)
	PurgeEmptyGroups	Remove empty groups from a grouping view. (Inherited from C1.LiveLinq.LiveViews.View)
	Rebuild	Re-populates the view by re-executing the view's query. (Inherited from C1.LiveLinq.LiveViews.View)
	Refresh	Loads the entities of the client view ignoring the client-side cache. (Inherited from C1.Data.ClientView<T>)
	Select<TResult>	Projects each element of a view into a new form. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	SelectMany	Overloaded. Projects each element of this view to a collection of collections, flattens the resulting collections into one collection, and invokes a result selector function on each element therein. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	SetTransaction	Sets the value of the C1.LiveLinq.LiveViews.View.Transaction property. (Inherited from C1.LiveLinq.LiveViews.View)
	ToString	Returns a string representing this view. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Union	Set union of two views. (Inherited from

		C1.LiveLinq.LiveViews.View<T>)
	Where	Filters the source view based on a predicate. (Inherited from C1.LiveLinq.LiveViews.View<T>)

Top

Public Events

	Name	Description
	Changed	Occurs after an item of the view or the entire view has changed. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Loaded	Occurs when the client view has finished loading succesfully, and also when an exception has been thrown during loading. (Inherited from C1.Data.ClientView<T>)

Top

See Also

Reference

PagingView<T> Class
C1.Data Namespace

Properties
For a list of all members of this type, see PagingView<T> members.

Public Properties

	Name	Description
	AutoLoad	Gets or sets a boolean value indicating whether the client view must be loaded automatically when its data is accessed. (Inherited from C1.Data.ClientView<T>)
	Count	Gets the total number of elements in the view. (Inherited from

		C1.LiveLinq.LiveViews.View)
	CurrentItem	Gets the current item in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	DataBindingMode	Gets or sets the data binding mode for this view. (Inherited from C1.LiveLinq.LiveViews.View)
	DeferredMaintenance	Gets the effective value of MaintenanceMode. (Inherited from C1.LiveLinq.LiveViews.View)
	Indexes	Gets the collection of indexes for this view. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	IsLoaded	Gets a value indicating whether the client view is loaded. (Inherited from C1.Data.ClientView<T>)
	IsLoading	Gets a value indicating whether the client view is being loaded. (Inherited from C1.Data.ClientView<T>)
	IsReadOnly	Gets a value indicating whether this view is read-only, not updatable. (Inherited from C1.LiveLinq.LiveViews.View)
	Item	Gets the view item (element) at the specified ordinal position. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	LoadSize	Gets or sets a value controling the number of entities to load in one batch.
	MaintenanceMode	Gets or sets a value controlling how the view is synchronized with changes in its base data. (Inherited from C1.LiveLinq.LiveViews.View)
	MoveToFirstOnReset	Gets or sets a value indicating that the first item must be made current after initial loading or reset (on any C1.LiveLinq.SourceChangeType.Reset notification) if current item was

		not set by other means. The default is True. (Inherited from C1.LiveLinq.LiveViews.View)
	Order	Gets a value indicating whether and how this view preserves item order if it exists in its base data source. (Inherited from C1.LiveLinq.LiveViews.View)
	PageCount	Gets the number of pages in this view.
	PageIndex	Gets or sets the index of the current page.
	PageSize	Gets or sets the number of items in a page.
	Scope	Gets the client scope to which this client view belongs. (Inherited from C1.Data.ClientView<T>)
	TotalItemCount	Gets the total number of entities in the view before paging is applied.
	Transaction	Gets an instance of C1.LiveLinq.ITransaction associated with the view. If a view has a transaction associated with it, that transaction's scope is opened automatically every time the view is updated, so the programmer does not need to do it manually in code. (Inherited from C1.LiveLinq.LiveViews.View)

Top

See Also

Reference

PagingView<T> Class
C1.Data Namespace

LoadSize Property
Gets or sets a value controling the number of entities to load in one batch.

Syntax

Visual Basic (Declaration)
`Public Property LoadSize As Integer`
C#
`public int LoadSize {get; set;}`

Remarks

Entities will be loaded using the multiple of PageSize nearest LoadSize. This allows multiple pages to be loaded at once without loading partial pages.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

PagingView<T> Class
PagingView<T> Members

PageCount Property
Gets the number of pages in this view.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property PageCount As Integer`
C#
`public int PageCount {get;}`

Remarks

If PageSize is 0, PageCount is also 0.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

PagingView<T> Class
PagingView<T> Members

PageIndex Property
Gets or sets the index of the current page.

Syntax

Visual Basic (Declaration)
`Public Property PageIndex As Integer`
C#
`public int PageIndex {get; set;}`

Remarks

Setting this property value to an invalid value is ignored. A value is invalid if it is less than 0 or greater or equal to PageCount. If there are no items in this view, the only valid value for this property is 0.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

PagingView<T> Class
PagingView<T> Members

PageSize Property
Gets or sets the number of items in a page.

Syntax

Visual Basic (Declaration)
`Public Property PageSize As Integer`
C#
`public int PageSize {get; set;}`

Remarks

To disable paging, set this property to 0.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

PagingView<T> Class
PagingView<T> Members

TotalItemCount Property
Gets the total number of entities in the view before paging is applied.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property TotalItemCount As Integer`
C#
`public int TotalItemCount {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

PagingView<T> Class
PagingView<T> Members

ProgressiveView<T>
The type of the entities in this view.

Represents a client view that loads entities sequentially (progressively) page by page. The user sees the result and can interact with it before all pages are loaded.

Object Model

Syntax

Visual Basic (Declaration)

`Public Class ProgressiveView(Of T)`
`  Inherits ClientView(Of T)`
`  Implements C1.LiveLinq.Indexing.IIndexedSource(Of T),`
`C1.LiveLinq.IObservableSource(Of T)`

C#
`public class ProgressiveView<T> : ClientView<T>,` `C1.LiveLinq.Indexing.IIndexedSource<T>, C1.LiveLinq.IObservableSource<T>`

Type Parameters

T

The type of the entities in this view.

Remarks

Sorting is required, loading entities progressively is impossible without sort.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.View
    C1.LiveLinq.LiveViews.View<T>
      C1.Data.ClientView<T>
        C1.Data.ProgressiveView<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ProgressiveView<T> Members
C1.Data Namespace

Overview
The type of the entities in this view.

Represents a client view that loads entities sequentially (progressively) page by page. The user sees the result and can interact with it before all pages are loaded.

Object Model

Syntax

Visual Basic (Declaration)	

`Public Class ProgressiveView(Of T)`
`  Inherits ClientView(Of T)`
`  Implements C1.LiveLinq.Indexing.IIndexedSource(Of T),`
`C1.LiveLinq.IObservableSource(Of T)`	

C#	

```
public class ProgressiveView<T> : ClientView<T>,
C1.LiveLinq.Indexing.IIndexedSource<T>, C1.LiveLinq.IObservableSource<T>
```

Type Parameters

T

The type of the entities in this view.

Remarks

Sorting is required, loading entities progressively is impossible without sort.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.View
    C1.LiveLinq.LiveViews.View<T>
      C1.Data.ClientView<T>
        C1.Data.ProgressiveView<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ProgressiveView<T> Members
C1.Data Namespace

Members
Properties Methods Events

The following tables list the members exposed by ProgressiveView<T>.

Public Properties

	Name	Description

	AutoLoad	Gets or sets a boolean value indicating whether the client view must be loaded automatically when its data is accessed. (Inherited from C1.Data.ClientView<T>)
	Count	Gets the total number of elements in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	CurrentItem	Gets the current item in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	DataBindingMode	Gets or sets the data binding mode for this view. (Inherited from C1.LiveLinq.LiveViews.View)
	DeferredMaintenance	Gets the effective value of MaintenanceMode. (Inherited from C1.LiveLinq.LiveViews.View)
	Indexes	Gets the collection of indexes for this view. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	IsLoaded	Gets a value indicating whether the client view is loaded. (Inherited from C1.Data.ClientView<T>)
	IsLoading	Gets a value indicating whether the client view is being loaded. (Inherited from C1.Data.ClientView<T>)
	IsReadOnly	Gets a value indicating whether this view is read-only, not updatable. (Inherited from C1.LiveLinq.LiveViews.View)
	Item	Gets the view item (element) at the specified ordinal position. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	LoadSize	Gets or sets the size of a page. To disable progressive loading, set this property to 0.
	MaintenanceMode	Gets or sets a value controlling how the view is synchronized with

		changes in its base data. (Inherited from C1.LiveLinq.LiveViews.View)
	MoveToFirstOnReset	Gets or sets a value indicating that the first item must be made current after initial loading or reset (on any C1.LiveLinq.SourceChangeType.Reset notification) if current item was not set by other means. The default is True. (Inherited from C1.LiveLinq.LiveViews.View)
	Order	Gets a value indicating whether and how this view preserves item order if it exists in its base data source. (Inherited from C1.LiveLinq.LiveViews.View)
	Scope	Gets the client scope to which this client view belongs. (Inherited from C1.Data.ClientView<T>)
	Transaction	Gets an instance of C1.LiveLinq.ITransaction associated with the view. If a view has a transaction associated with it, that transaction's scope is opened automatically every time the view is updated, so the programmer does not need to do it manually in code. (Inherited from C1.LiveLinq.LiveViews.View)

Top

Public Methods

	Name	Description
	AsCollectionViewFactory	Returns an instance of System.ComponentModel.ICollectionViewFactory that can be used as a source of a System.Windows.Data.CollectionViewSource. (Inherited from C1.LiveLinq.LiveViews.View)
	AsDynamic	Used for views with anonymous type constructor as the result selector, converts the C1.LiveLinq.LiveViews.View to a View<dynamic> so it can be used for data binding and

		programmatic access. (Inherited from C1.LiveLinq.LiveViews.View)
	AsFiltered	Filters the view on the server side using a predicate. (Inherited from C1.Data.ClientView<T>)
	AsFilteredBound<TKey>	Filters the view on the server using a key selector function and configurable value and operator. (Inherited from C1.Data.ClientView<T>)
	AttachAggregationView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	AttachView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	CancelLoad	Cancels the current loading operation. (Inherited from C1.Data.ClientView<T>)
	Concat	Concatenation of two views. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Contains	Determines whether the view contains a specified item. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	DeferMaintenance	Enters a defer cycle that you can use to make bulk changes to the view sources and delay automatic view maintenance. (Inherited from C1.LiveLinq.LiveViews.View)
	GetEnumerator	Returns an enumerator that iterates through the view items. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	GroupBy	Overloaded. Groups the elements of a view according to a specified key selector function. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	GroupJoin<TInner,TKey,TResult>	Correlates the elements of two views based on equality of

		keys and groups the results. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Include	Specifies related objects to include while loading the client view. (Inherited from C1.Data.ClientView<T>)
	IndexOf	Searches for the specified object among the elements of the view and returns the zero-based ordinal position of its first occurrence. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Join<TInner,TKey,TResult>	Correlates the elements of two views based on matching keys. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Load	Loads the entities of the client view. (Inherited from C1.Data.ClientView<T>)
	Maintain	Brings the view up to date with its source data. (Inherited from C1.LiveLinq.LiveViews.View)
	OrderBy<TKey>	Sorts the elements of a view in ascending order. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	OrderByDescending<TKey>	Sorts the elements of a view in descending order. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Paging	Overloaded. Applies paging to this client view. (Inherited from C1.Data.ClientView<T>)
	ProgressiveLoading	Overloaded. Specifies that the client view loading is performed not in a single trip to the server but in multiple batches, each loading a page of a limited size so the user sees the result and can interact with it before all pages are loaded. (Inherited from C1.Data.ClientView<T>)

	PurgeEmptyGroups	Remove empty groups from a grouping view. (Inherited from C1.LiveLinq.LiveViews.View)
	Rebuild	Re-populates the view by re-executing the view's query. (Inherited from C1.LiveLinq.LiveViews.View)
	Refresh	Loads the entities of the client view ignoring the client-side cache. (Inherited from C1.Data.ClientView<T>)
	Select<TResult>	Projects each element of a view into a new form. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	SelectMany	Overloaded. Projects each element of this view to a collection of collections, flattens the resulting collections into one collection, and invokes a result selector function on each element therein. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	SetTransaction	Sets the value of the C1.LiveLinq.LiveViews.View.Transaction property. (Inherited from C1.LiveLinq.LiveViews.View)
	ToString	Returns a string representing this view. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Union	Set union of two views. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Where	Filters the source view based on a predicate. (Inherited from C1.LiveLinq.LiveViews.View<T>)

Top

Public Events

	Name	Description

	Changed	Occurs after an item of the view or the entire view has changed. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Loaded	Occurs when the client view has finished loading succesfully, and also when an exception has been thrown during loading. (Inherited from C1.Data.ClientView<T>)

Top

See Also

Reference

ProgressiveView<T> Class
C1.Data Namespace

Properties
For a list of all members of this type, see ProgressiveView<T> members.

Public Properties

	Name	Description
	AutoLoad	Gets or sets a boolean value indicating whether the client view must be loaded automatically when its data is accessed. (Inherited from C1.Data.ClientView<T>)
	Count	Gets the total number of elements in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	CurrentItem	Gets the current item in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	DataBindingMode	Gets or sets the data binding mode for this view. (Inherited from C1.LiveLinq.LiveViews.View)
	DeferredMaintenance	Gets the effective value of MaintenanceMode. (Inherited from C1.LiveLinq.LiveViews.View)

	Indexes	Gets the collection of indexes for this view. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	IsLoaded	Gets a value indicating whether the client view is loaded. (Inherited from C1.Data.ClientView<T>)
	IsLoading	Gets a value indicating whether the client view is being loaded. (Inherited from C1.Data.ClientView<T>)
	IsReadOnly	Gets a value indicating whether this view is read-only, not updatable. (Inherited from C1.LiveLinq.LiveViews.View)
	Item	Gets the view item (element) at the specified ordinal position. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	LoadSize	Gets or sets the size of a page. To disable progressive loading, set this property to 0.
	MaintenanceMode	Gets or sets a value controlling how the view is synchronized with changes in its base data. (Inherited from C1.LiveLinq.LiveViews.View)
	MoveToFirstOnReset	Gets or sets a value indicating that the first item must be made current after initial loading or reset (on any C1.LiveLinq.SourceChangeType.Reset notification) if current item was not set by other means. The default is True. (Inherited from C1.LiveLinq.LiveViews.View)
	Order	Gets a value indicating whether and how this view preserves item order if it exists in its base data source. (Inherited from C1.LiveLinq.LiveViews.View)
	Scope	Gets the client scope to which this client view belongs. (Inherited from C1.Data.ClientView<T>)

	Transaction	Gets an instance of C1.LiveLinq.ITransaction associated with the view. If a view has a transaction associated with it, that transaction's scope is opened automatically every time the view is updated, so the programmer does not need to do it manually in code. (Inherited from C1.LiveLinq.LiveViews.View)

Top

See Also

Reference

ProgressiveView<T> Class
C1.Data Namespace

LoadSize Property
Gets or sets the size of a page. To disable progressive loading, set this property to 0.

Syntax

Visual Basic (Declaration)

```
Public Property LoadSize As Integer
```

C#

```
public int LoadSize {get; set;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ProgressiveView<T> Class
ProgressiveView<T> Members

SavedChangesEventArgs
Provides data for the C1DataSource.SavedChanges event.

Object Model

SavedChangesEventArgs

Syntax

Visual Basic (Declaration)
`Public Class SavedChangesEventArgs` `  Inherits System.EventArgs`
C#
`public class SavedChangesEventArgs : System.EventArgs`

Inheritance Hierarchy

System.Object
  System.EventArgs
    C1.Data.SavedChangesEventArgs

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SavedChangesEventArgs Members
C1.Data Namespace

Overview
Provides data for the C1DataSource.SavedChanges event.

Object Model

SavedChangesEventArgs

Syntax

Visual Basic (Declaration)
`Public Class SavedChangesEventArgs` `  Inherits System.EventArgs`
C#
`public class SavedChangesEventArgs : System.EventArgs`

Inheritance Hierarchy

System.Object
  System.EventArgs
    C1.Data.SavedChangesEventArgs

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SavedChangesEventArgs Members
C1.Data Namespace

Members
Properties Methods

The following tables list the members exposed by SavedChangesEventArgs.

Public Properties

	Name	Description
	Error	Gets a value showing the error that occurred during a save operation.
	HasError	Gets a value indicating whether the save operation has failed. If true, inspect the Error property for details.

	IsErrorHandled	Gets a value indicating whether the error has been marked as handled by calling MarkErrorAsHandled.

Top

Public Methods

	Name	Description
	MarkErrorAsHandled	If this method is called for a failed operation (if HasError is true), it marks the error as handled. Otherwise, an exception is thrown.

Top

See Also

Reference

SavedChangesEventArgs Class
C1.Data Namespace

Methods
For a list of all members of this type, see SavedChangesEventArgs members.

Public Methods

	Name	Description
	MarkErrorAsHandled	If this method is called for a failed operation (if HasError is true), it marks the error as handled. Otherwise, an exception is thrown.

Top

See Also

Reference

SavedChangesEventArgs Class
C1.Data Namespace

MarkErrorAsHandled Method

If this method is called for a failed operation (if HasError is true), it marks the error as handled. Otherwise, an exception is thrown.

Syntax

Visual Basic (Declaration)
`Public Sub MarkErrorAsHandled()`
C#
`public void MarkErrorAsHandled()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SavedChangesEventArgs Class
SavedChangesEventArgs Members

Properties

For a list of all members of this type, see SavedChangesEventArgs members.

Public Properties

	Name	Description
	Error	Gets a value showing the error that occurred during a save operation.
	HasError	Gets a value indicating whether the save operation has failed. If true, inspect the Error property for details.
	IsErrorHandled	Gets a value indicating whether the error has been marked as handled by calling MarkErrorAsHandled.

Top

See Also

Reference

SavedChangesEventArgs Class
C1.Data Namespace

Error Property
Gets a value showing the error that occurred during a save operation.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Error As Exception`
C#
`public Exception Error {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SavedChangesEventArgs Class
SavedChangesEventArgs Members

HasError Property
Gets a value indicating whether the save operation has failed. If true, inspect the Error property for details.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property HasError As Boolean`

C#

```
public bool HasError {get;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SavedChangesEventArgs Class
SavedChangesEventArgs Members

IsErrorHandled Property
Gets a value indicating whether the error has been marked as handled by calling MarkErrorAsHandled.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Property IsErrorHandled As Boolean
```

C#

```
public bool IsErrorHandled {get;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SavedChangesEventArgs Class
SavedChangesEventArgs Members

C1.Data.DataSource Namespace

Overview

Classes

	Class	Description
	BaseControlHandler	A base class for control handlers that connect GUI controls of supported types to a C1DataSource so that those controls can be given additional functionality such as auto-lookup columns and virtual mode.
	ClientCollectionView	The collection view implementation used by a ClientViewSource and other Studio for Entity Framework data sources.
	ClientViewSource	Data source object exposing data from C1.Data.ClientCacheBase to which GUI controls can bind. Using a ClientViewSource, you can load, filter, group, and sort data easily.
	ClientViewSourceException	This exception indicates that a ClientViewSource is miconfigured or an error has occurred during the ClientViewSource.Load operation.
	DependencyObjectCollection<T>	An observable collection of dependency objects.
	FilterDescriptor	Descriptor used by the ClientViewSource to filter data in queries.
	FilterDescriptorCollection	Collection of FilterDescriptor dependency objects.
	GroupDescriptor	Descriptor used by the ClientViewSource to group data returned from server-side queries.
	GroupDescriptorCollection	Collection of GroupDescriptor dependency objects.

	SortDescriptor	Descriptor used by the ClientViewSource to sort data returned from queries.
	SortDescriptorCollection	Collection of SortDescriptor dependency objects.

Enumerations

	Enumeration	Description
	FilterDescriptorLogicalOperator	Enumeration of logical operators for filter collections.
	FilterOperator	Operator used in FilterDescriptor class.
	VirtualModeKind	Enumeration of possible virtual modes a ClientViewSource can be in. Used in the ClientViewSource.VirtualMode property.

See Also

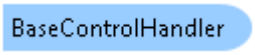
Reference

C1.Data.Entity.4 Assembly

Classes

BaseControlHandler
A base class for control handlers that connect GUI controls of supported types to a C1DataSource so that those controls can be given additional functionality such as auto-lookup columns and virtual mode.

Object Model



Syntax

Visual Basic (Declaration)	

```
Public MustInherit Class BaseControlHandler
   Inherits System.Windows.DependencyObject
```

C#

```
public abstract class BaseControlHandler : System.Windows.DependencyObject
```

Remarks

Use platform-specific control handlers for your controls: C1.Win.Data.ControlHandler, C1.WPF.Data.ControlHandler, and C1.Silverlight.Data.ControlHandler.

The list of supported GUI controls for each platform can be found in the reference of that platform's ControlHandler.

Inheritance Hierarchy

System.Object
  System.Windows.Threading.DispatcherObject
    System.Windows.DependencyObject
      C1.Data.DataSource.BaseControlHandler
        C1.Silverlight.Data.ControlHandler
        C1.Win.Data.ControlHandler
        C1.WPF.Data.ControlHandler

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

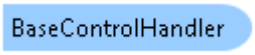
See Also

Reference

BaseControlHandler Members
C1.Data.DataSource Namespace

Overview
A base class for control handlers that connect GUI controls of supported types to a C1DataSource so that those controls can be given additional functionality such as auto-lookup columns and virtual mode.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit Class BaseControlHandler` `  Inherits System.Windows.DependencyObject`
C#
`public abstract class BaseControlHandler : System.Windows.DependencyObject`

Remarks

Use platform-specific control handlers for your controls: C1.Win.Data.ControlHandler, C1.WPF.Data.ControlHandler, and C1.Silverlight.Data.ControlHandler.

The list of supported GUI controls for each platform can be found in the reference of that platform's ControlHandler.

Inheritance Hierarchy

System.Object
  System.Windows.Threading.DispatcherObject
    System.Windows.DependencyObject
      C1.Data.DataSource.BaseControlHandler
        C1.Silverlight.Data.ControlHandler
        C1.Win.Data.ControlHandler
        C1.WPF.Data.ControlHandler

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

BaseControlHandler Members
C1.Data.DataSource Namespace

Members
Fields Properties Methods

The following tables list the members exposed by BaseControlHandler.

Public Fields

	Name	Description
S	AutoLookupProperty	The DependencyProperty for the AutoLookup property.
S	VirtualModeProperty	The DependencyProperty for the VirtualMode property.

Top

Public Properties

	Name	Description
	AutoLookup	Gets or sets a value indicating whether data grid columns bound to navigation (foreign key, lookup) properties must be converted to combo box columns, so the user can see the right value and edit it by choosing a value from a drop-down list. The default value is False.
	DependencyObjectType	(Inherited from System.Windows.DependencyObject)
	Dispatcher	(Inherited from System.Windows.Threading.DispatcherObject)
	IsSealed	(Inherited from System.Windows.DependencyObject)
	SupportsVirtualMode	Gets a value indicating whether this control handler supports Virtual Mode.
	VirtualMode	Gets or sets a value indicating whether virtual mode specified in a ClientViewSource is managed by this control handler.

Top

Public Methods

	Name	Description
	Apply	Forces this control handler to apply its settings to the current control.
	ClearValue	Overloaded. (Inherited from System.Windows.DependencyObject)
	CoerceValue	(Inherited from System.Windows.DependencyObject)
	Equals	(Inherited from System.Windows.DependencyObject)
	GetHashCode	(Inherited from System.Windows.DependencyObject)
	GetLocalValueEnumerator	(Inherited from System.Windows.DependencyObject)
	GetValue	(Inherited from System.Windows.DependencyObject)
	InvalidateProperty	(Inherited from System.Windows.DependencyObject)
	ReadLocalValue	(Inherited from System.Windows.DependencyObject)
	SetCurrentValue	(Inherited from System.Windows.DependencyObject)
	SetValue	Overloaded. (Inherited from System.Windows.DependencyObject)

Top

See Also

Reference

BaseControlHandler Class
C1.Data.DataSource Namespace

Methods
For a list of all members of this type, see BaseControlHandler members.

Public Methods

	Name	Description
	Apply	Forces this control handler to apply its settings to the current control.
	ClearValue	Overloaded. (Inherited from System.Windows.DependencyObject)
	CoerceValue	(Inherited from System.Windows.DependencyObject)
	Equals	(Inherited from System.Windows.DependencyObject)
	GetHashCode	(Inherited from System.Windows.DependencyObject)
	GetLocalValueEnumerator	(Inherited from System.Windows.DependencyObject)
	GetValue	(Inherited from System.Windows.DependencyObject)
	InvalidateProperty	(Inherited from System.Windows.DependencyObject)
	ReadLocalValue	(Inherited from System.Windows.DependencyObject)
	SetCurrentValue	(Inherited from System.Windows.DependencyObject)
	SetValue	Overloaded. (Inherited from System.Windows.DependencyObject)

Top

See Also

Reference

BaseControlHandler Class
C1.Data.DataSource Namespace

Apply Method
Forces this control handler to apply its settings to the current control.

Syntax

Visual Basic (Declaration)	
`Public Overridable Sub Apply()`	
C#	
`public virtual void Apply()`	

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

BaseControlHandler Class
BaseControlHandler Members

Properties
For a list of all members of this type, see BaseControlHandler members.

Public Properties

	Name	Description
	AutoLookup	Gets or sets a value indicating whether data grid columns bound to navigation (foreign key, lookup) properties must be converted to combo box columns, so the user can see the right value and edit it by choosing a value from a drop-down list. The default value is False.
	DependencyObjectType	(Inherited from System.Windows.DependencyObject)
	Dispatcher	(Inherited from System.Windows.Threading.DispatcherObject)
	IsSealed	(Inherited from System.Windows.DependencyObject)
	SupportsVirtualMode	Gets a value indicating whether this control handler supports Virtual

		Mode.
	VirtualMode	Gets or sets a value indicating whether virtual mode specified in a ClientViewSource is managed by this control handler.

Top

See Also

Reference

BaseControlHandler Class
C1.Data.DataSource Namespace

AutoLookup Property
Gets or sets a value indicating whether data grid columns bound to navigation (foreign key, lookup) properties must be converted to combo box columns, so the user can see the right value and edit it by choosing a value from a drop-down list. The default value is False.

Syntax

Visual Basic (Declaration)

```
Public Property AutoLookup As Boolean
```

C#

```
public bool AutoLookup {get; set;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

BaseControlHandler Class
BaseControlHandler Members

SupportsVirtualMode Property
Gets a value indicating whether this control handler supports Virtual Mode.

Syntax

Visual Basic (Declaration)
`Public Overridable ReadOnly Property SupportsVirtualMode As Boolean`
C#
`public virtual bool SupportsVirtualMode {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

BaseControlHandler Class
BaseControlHandler Members
VirtualMode Property

VirtualMode Property
Gets or sets a value indicating whether virtual mode specified in a ClientViewSource is managed by this control handler.

Syntax

Visual Basic (Declaration)
`Public Property VirtualMode As Boolean`
C#
`public bool VirtualMode {get; set;}`

Remarks

Setting this property to True has effect only if ClientViewSource.VirtualMode of the associated ClientViewSource is set to Managed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

BaseControlHandler Class
BaseControlHandler Members

Fields
For a list of all members of this type, see BaseControlHandler members.

Public Fields

	Name	Description
S	AutoLookupProperty	The DependencyProperty for the AutoLookup property.
S	VirtualModeProperty	The DependencyProperty for the VirtualMode property.

Top

See Also

Reference

BaseControlHandler Class
C1.Data.DataSource Namespace

AutoLookupProperty Field
The DependencyProperty for the AutoLookup property.

Syntax

Visual Basic (Declaration)
`Public Shared ReadOnly AutoLookupProperty As DependencyProperty`
C#
`public static readonly DependencyProperty AutoLookupProperty`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

BaseControlHandler Class
BaseControlHandler Members

VirtualModeProperty Field
The DependencyProperty for the VirtualMode property.

Syntax

Visual Basic (Declaration)
`Public Shared ReadOnly VirtualModeProperty As DependencyProperty`
C#
`public static readonly DependencyProperty VirtualModeProperty`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

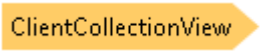
See Also

Reference

BaseControlHandler Class
BaseControlHandler Members

ClientCollectionView
The collection view implementation used by a ClientViewSource and other Studio for Entity Framework data sources.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class ClientCollectionView`
C#
`public class ClientCollectionView`

Inheritance Hierarchy

System.Object
  C1.Data.DataSource.ClientCollectionView

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

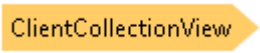
See Also

Reference

ClientCollectionView Members
C1.Data.DataSource Namespace

Overview
The collection view implementation used by a ClientViewSource and other Studio for Entity Framework data sources.

Object Model



Syntax

Visual Basic (Declaration)

```
Public Class ClientCollectionView
```

C#

```
public class ClientCollectionView
```

Inheritance Hierarchy

System.Object
  C1.Data.DataSource.ClientCollectionView

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Members
C1.Data.DataSource Namespace

Members
Properties Methods Events

The following tables list the members exposed by ClientCollectionView.

Public Properties

	Name	Description
	CanAdd	Gets a value indicating whether the Add method is supported.
	CanChangePage	Gets a value that indicates whether the PageIndex value can change.
	CanRemove	Gets a value that indicates whether an item can be removed from the collection.
	CollectionViewFactory	Gets an instance of System.ComponentModel.ICollectionViewFactory that can be used as a source of a

		System.Windows.Data.CollectionViewSource.
	Count	Gets the number of elements contained in the ClientCollectionView.
	CurrentItem	Gets the current item in the view.
	CurrentPosition	Gets the ordinal position of the CurrentItem within the collection view.
	IsEmpty	Returns a value that indicates whether the resulting view is empty.
	IsPageChanging	Gets a value that indicates whether the page index is changing.
	Item	Gets the element at the specified index.
	ItemProperties	Gets a collection that contains information about the properties that are available on the items in this ClientCollectionView.
	PageCount	Gets the count of the pages in this view.
	PageIndex	Gets the zero-based index of the current page.
	PageSize	Gets or sets the number of items to display on a page.
	TotalItemCount	Gets the total number of items in the view before paging is applied.

Top

Public Methods

	Name	Description
	Add	Adds a new entity to the client-side cache and to the associated context. The entity will appear in this collection view if it matches the underlying query.
	AsLive<T>	Converts this ClientCollectionView to a live view.

	Contains	Returns a value that indicates whether a given item belongs to this collection view.
	IndexOf	Determines the index of a specific item in the ClientCollectionView.
	MoveCurrentTo	Sets the specified item to be the CurrentItem in the view.
	MoveCurrentToFirst	Sets the first item in the view as the CurrentItem.
	MoveCurrentToLast	Sets the last item in the view as the CurrentItem.
	MoveCurrentToNext	Sets the item after the CurrentItem in the view as the System.ComponentModel.ICollectionView.CurrentItem.
	MoveCurrentToPosition	Sets the item at the specified index to be the CurrentItem in the view.
	MoveCurrentToPrevious	Sets the item before the CurrentItem in the view as the CurrentItem.
	MoveToFirstPage	Sets the first page as the current page.
	MoveToLastPage	Sets the last page as the current page.
	MoveToNextPage	Moves to the page after the current page.
	MoveToPage	Sets the first page as the current page.
	MoveToPreviousPage	Moves to the page before the current page.
	Remove	Removes the specified item from the collection.
	RemoveAt	Removes the item at the specified position from the collection.

Top

Public Events

	Name	Description
	CurrentChanged	When implementing this interface, raise this event after the current item has been changed.
	CurrentChanging	When implementing this interface, raise this event before changing the current item. Event handler can cancel this event.
	PageChanged	Occurs after the PageIndex has changed.
	PageChanging	Occurs before changing the PageIndex.
	PropertyChanged	Occurs when a property value changes.

Top

See Also

Reference

ClientCollectionView Class
C1.Data.DataSource Namespace

Methods
For a list of all members of this type, see ClientCollectionView members.

Public Methods

	Name	Description
	Add	Adds a new entity to the client-side cache and to the associated context. The entity will appear in this collection view if it matches the underlying query.
	AsLive<T>	Converts this ClientCollectionView to a live view.
	Contains	Returns a value that indicates whether a given item belongs to this collection view.

	IndexOf	Determines the index of a specific item in the ClientCollectionView.
	MoveCurrentTo	Sets the specified item to be the CurrentItem in the view.
	MoveCurrentToFirst	Sets the first item in the view as the CurrentItem.
	MoveCurrentToLast	Sets the last item in the view as the CurrentItem.
	MoveCurrentToNext	Sets the item after the CurrentItem in the view as the System.ComponentModel.ICollectionView.CurrentItem.
	MoveCurrentToPosition	Sets the item at the specified index to be the CurrentItem in the view.
	MoveCurrentToPrevious	Sets the item before the CurrentItem in the view as the CurrentItem.
	MoveToFirstPage	Sets the first page as the current page.
	MoveToLastPage	Sets the last page as the current page.
	MoveToNextPage	Moves to the page after the current page.
	MoveToPage	Sets the first page as the current page.
	MoveToPreviousPage	Moves to the page before the current page.
	Remove	Removes the specified item from the collection.
	RemoveAt	Removes the item at the specified position from the collection.

Top

See Also

Reference

ClientCollectionView Class
C1.Data.DataSource Namespace

Add Method

The new entity to add.

Adds a new entity to the client-side cache and to the associated context. The entity will appear in this collection view if it matches the underlying query.

Syntax

Visual Basic (Declaration)

`Public Sub Add( _`
`  ByVal entity As Object _`
`)`

C#

`public void Add(`
`  object entity`
`)`

Parameters

entity

The new entity to add.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

AsLive<T> Method

The type of the elements in this collection view.

Converts this ClientCollectionView to a live view.

Syntax

Visual Basic (Declaration)
`Public Function AsLive(Of T)() As View(Of T)`
C#
`public View<T> AsLive<T>()`

Type Parameters

T

The type of the elements in this collection view.

Return Value

The resulting live view.

Exceptions

Exception	Description
System.NotSupportedException	The ClientViewSource is in virtual mode.

Remarks

This method does not change the ClientCollectionView in any way, it just exposes its live view functionality.

This method is not supported for a ClientViewSource in virtual mode.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

Contains Method

The object to check.

Returns a value that indicates whether a given item belongs to this collection view.

Syntax

Visual Basic (Declaration)

`Public Function Contains( _`
`  ByVal item As Object _`
`) As Boolean`

C#

`public bool Contains(`
`  object item`
`)`

Parameters

item

The object to check.

Return Value

true if the item belongs to this collection view; otherwise, false.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

IndexOf Method

The item to locate in the ClientCollectionView.

Determines the index of a specific item in the ClientCollectionView.

Syntax

Visual Basic (Declaration)

```
Public Function IndexOf( _
   ByVal item As Object _
) As Integer
```

C#

```
public int IndexOf(
   object item
)
```

Parameters

item

The item to locate in the ClientCollectionView.

Return Value

The index of the item if found in the list; otherwise, -1.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

MoveCurrentTo Method
The item to set as the CurrentItem.

Sets the specified item to be the CurrentItem in the view.

Syntax

Visual Basic (Declaration)

```
Public Function MoveCurrentTo( _
   ByVal item As Object _
) As Boolean
```

C#

```
public bool MoveCurrentTo(
   object item
)
```

Parameters

item

The item to set as the CurrentItem.

Return Value

true if the resulting CurrentItem is within the view; otherwise, false.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

MoveCurrentToFirst Method

Sets the first item in the view as the CurrentItem.

Syntax

Visual Basic (Declaration)

```
Public Function MoveCurrentToFirst() As Boolean
```

C#
`public bool MoveCurrentToFirst()`

Return Value

true if the resulting CurrentItem is an item within the view; otherwise, false.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

MoveCurrentToLast Method
Sets the last item in the view as the CurrentItem.

Syntax

Visual Basic (Declaration)
`Public Function MoveCurrentToLast() As Boolean`
C#
`public bool MoveCurrentToLast()`

Return Value

true if the resulting CurrentItem is an item within the view; otherwise, false.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

MoveCurrentToNext Method
Sets the item after the CurrentItem in the view as the System.ComponentModel.ICollectionView.CurrentItem.

Syntax

Visual Basic (Declaration)
`Public Function MoveCurrentToNext() As Boolean`
C#
`public bool MoveCurrentToNext()`

Return Value

true if the resulting CurrentItem is an item within the view; otherwise, false.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

MoveCurrentToPosition Method
The index to set the System.ComponentModel.ICollectionView.CurrentItem to.

Sets the item at the specified index to be the CurrentItem in the view.

Syntax

Visual Basic (Declaration)

```
Public Function MoveCurrentToPosition( _
   ByVal position As Integer _
) As Boolean
```

C#

```
public bool MoveCurrentToPosition(
   int position
)
```

Parameters

position

The index to set the System.ComponentModel.ICollectionView.CurrentItem to.

Return Value

true if the resulting CurrentItem is an item within the view; otherwise, false.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

MoveCurrentToPrevious Method

Sets the item before the CurrentItem in the view as the CurrentItem.

Syntax

Visual Basic (Declaration)

```
Public Function MoveCurrentToPrevious() As Boolean
```

C#

```
public bool MoveCurrentToPrevious()
```

Return Value

true if the resulting CurrentItem is an item within the view; otherwise, false.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

MoveToFirstPage Method
Sets the first page as the current page.

Syntax

Visual Basic (Declaration)
`Public Function MoveToFirstPage() As Boolean`
C#
`public bool MoveToFirstPage()`

Return Value

true if the operation was successful; otherwise, false.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

MoveToLastPage Method
Sets the last page as the current page.

Syntax

Visual Basic (Declaration)
`Public Function MoveToLastPage() As Boolean`
C#
`public bool MoveToLastPage()`

Return Value

true if the operation was successful; otherwise, false.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

MoveToNextPage Method
Moves to the page after the current page.

Syntax

Visual Basic (Declaration)
`Public Function MoveToNextPage() As Boolean`
C#
`public bool MoveToNextPage()`

Return Value

true if the operation was successful; otherwise, false.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

MoveToPage Method
The index of the page to move to.

Sets the first page as the current page.

Syntax

Visual Basic (Declaration)	

`Public Function MoveToPage( _`
`  ByVal pageIndex As Integer _`
`) As Boolean`	

C#	

`public bool MoveToPage(`
`  int pageIndex`
`)`	

Parameters

pageIndex

The index of the page to move to.

Return Value

True if the operation was successful; otherwise, False.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

MoveToPreviousPage Method
Moves to the page before the current page.

Syntax

Visual Basic (Declaration)
`Public Function MoveToPreviousPage() As Boolean`
C#
`public bool MoveToPreviousPage()`

Return Value

true if the operation was successful; otherwise, false.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

Remove Method
The item to remove.

Removes the specified item from the collection.

Syntax

Visual Basic (Declaration)

```
Public Sub Remove( _
   ByVal item As Object _
)
```

C#

```
public void Remove(
   object item
)
```

Parameters

item

The item to remove.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

RemoveAt Method

The position of the item to remove.

Removes the item at the specified position from the collection.

Syntax

Visual Basic (Declaration)

```
Public Sub RemoveAt( _
   ByVal index As Integer _
)
```

C#

```
public void RemoveAt(
   int index
)
```

Parameters

index

The position of the item to remove.

Exceptions

Exception	Description
System.ArgumentOutOfRangeException	index is less than 0 or greater than the number of items in the collection view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

Properties

For a list of all members of this type, see ClientCollectionView members.

Public Properties

	Name	Description

	CanAdd	Gets a value indicating whether the Add method is supported.
	CanChangePage	Gets a value that indicates whether the PageIndex value can change.
	CanRemove	Gets a value that indicates whether an item can be removed from the collection.
	CollectionViewFactory	Gets an instance of System.ComponentModel.ICollectionViewFactory that can be used as a source of a System.Windows.Data.CollectionViewSource.
	Count	Gets the number of elements contained in the ClientCollectionView.
	CurrentItem	Gets the current item in the view.
	CurrentPosition	Gets the ordinal position of the CurrentItem within the collection view.
	IsEmpty	Returns a value that indicates whether the resulting view is empty.
	IsPageChanging	Gets a value that indicates whether the page index is changing.
	Item	Gets the element at the specified index.
	ItemProperties	Gets a collection that contains information about the properties that are available on the items in this ClientCollectionView.
	PageCount	Gets the count of the pages in this view.
	PageIndex	Gets the zero-based index of the current page.
	PageSize	Gets or sets the number of items to display on a page.
	TotalItemCount	Gets the total number of items in the view before paging is applied.

Top

See Also

Reference

ClientCollectionView Class
C1.Data.DataSource Namespace

CanAdd Property
Gets a value indicating whether the Add method is supported.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property CanAdd As Boolean`
C#
`public bool CanAdd {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

CanChangePage Property
Gets a value that indicates whether the PageIndex value can change.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property CanChangePage As Boolean`
C#

```
public bool CanChangePage {get;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

CanRemove Property
Gets a value that indicates whether an item can be removed from the collection.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property CanRemove As Boolean`
C#
`public bool CanRemove {get;}`

Property Value

true if an item can be removed from the collection; otherwise, false.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

CollectionViewFactory Property

Gets an instance of System.ComponentModel.ICollectionViewFactory that can be used as a source of a System.Windows.Data.CollectionViewSource.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property CollectionViewFactory As ICollectionViewFactory`
C#
`public ICollectionViewFactory CollectionViewFactory {get;}`

Property Value

A factory that returns the same ClientCollectionView as an System.ComponentModel.ICollectionView.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

Count Property

Gets the number of elements contained in the ClientCollectionView.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Count As Integer`
C#
`public int Count {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

CurrentItem Property
Gets the current item in the view.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property CurrentItem As Object`
C#
`public object CurrentItem {get;}`

Property Value

The current item of the view or null if there is no current item.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

CurrentPosition Property
Gets the ordinal position of the CurrentItem within the collection view.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property CurrentPosition As Integer`
C#
`public int CurrentPosition {get;}`

Property Value

The ordinal position of the CurrentItem within the collection view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

IsEmpty Property
Returns a value that indicates whether the resulting view is empty.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property IsEmpty As Boolean`
C#
`public bool IsEmpty {get;}`

Property Value

true if the resulting view is empty; otherwise, false.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

IsPageChanging Property
Gets a value that indicates whether the page index is changing.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property IsPageChanging As Boolean`
C#
`public bool IsPageChanging {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

Item Property
The zero-based index of the element to get.

Gets the element at the specified index.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Default Property Item( _
   ByVal index As Integer _
) As Object
```

C#

```
public object this[
   int index
]; {get;}
```

Parameters

index

The zero-based index of the element to get.

Property Value

The element at the specified index.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

ItemProperties Property
Gets a collection that contains information about the properties that are available on the items in this ClientCollectionView.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Property ItemProperties As ReadOnlyCollection(Of
```

```
ItemPropertyInfo)
```

C#

```
public ReadOnlyCollection<ItemPropertyInfo> ItemProperties {get;}
```

Property Value

A collection that contains information about the properties that are available on the items in this ClientCollectionView.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

PageCount Property
Gets the count of the pages in this view.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Property PageCount As Integer
```

C#

```
public int PageCount {get;}
```

Remarks

When PageSize is 0, the PageIndex will also be 0.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

PageIndex Property
Gets the zero-based index of the current page.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property PageIndex As Integer`
C#
`public int PageIndex {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

PageSize Property
Gets or sets the number of items to display on a page.

Syntax

Visual Basic (Declaration)

```
Public Property PageSize As Integer
```

C#

```
public int PageSize {get; set;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

TotalItemCount Property
Gets the total number of items in the view before paging is applied.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Property TotalItemCount As Integer
```

C#

```
public int TotalItemCount {get;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

Events

>

Name	Description
CurrentChanged	When implementing this interface, raise this event after the current item has been changed.
CurrentChanging	When implementing this interface, raise this event before changing the current item. Event handler can cancel this event.
PageChanged	Occurs after the PageIndex has changed.
PageChanging	Occurs before changing the PageIndex.
PropertyChanged	Occurs when a property value changes.

Top

See Also

Reference

ClientCollectionView Class
C1.Data.DataSource Namespace

CurrentChanged Event
When implementing this interface, raise this event after the current item has been changed.

Syntax

Visual Basic (Declaration)
`Public Event CurrentChanged As EventHandler`
C#
`public event EventHandler CurrentChanged`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

CurrentChanging Event
When implementing this interface, raise this event before changing the current item. Event handler can cancel this event.

Syntax

Visual Basic (Declaration)	
`Public Event CurrentChanging As CurrentChangingEventHandler`	
C#	
`public event CurrentChangingEventHandler CurrentChanging`	

Event Data

The event handler receives an argument of type CurrentChangingEventArgs containing data related to this event. The following **CurrentChangingEventArgs** properties provide information specific to this event.

Property	Description
Cancel	Gets or sets a value that indicates whether to cancel the event.
IsCancelable	Gets a value that indicates whether the event is cancelable.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

PageChanged Event
Occurs after the PageIndex has changed.

Syntax

Visual Basic (Declaration)
`Public Event PageChanged As EventHandler(Of EventArgs)`
C#
`public event EventHandler<EventArgs> PageChanged`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

PageChanging Event
Occurs before changing the PageIndex.

Syntax

Visual Basic (Declaration)
`Public Event PageChanging As EventHandler(Of PageChangingEventArgs)`
C#
`public event EventHandler<PageChangingEventArgs> PageChanging`

Event Data

The event handler receives an argument of type PageChangingEventArgs containing data related to this event. The following **PageChangingEventArgs** properties provide information specific to this event.

Property	Description
Cancel (Inherited from System.ComponentModel.CancelEventArgs)	
NewPageIndex	Gets the index of the requested page.

Remarks

This event allows to cancel the ongoing PageIndex change by setting System.ComponentModel.CancelEventArgs.Cancel to true.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

PropertyChanged Event
Occurs when a property value changes.

Syntax

Visual Basic (Declaration)
`Public Event PropertyChanged As PropertyChangedEventHandler`
C#
`public event PropertyChangedEventHandler PropertyChanged`

Event Data

The event handler receives an argument of type PropertyChangedEventArgs containing data related to this event. The following **PropertyChangedEventArgs** properties provide information specific to this event.

Property	Description
PropertyName	

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

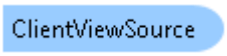
See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

ClientViewSource
Data source object exposing data from C1.Data.ClientCacheBase to which GUI controls can bind. Using a ClientViewSource, you can load, filter, group, and sort data easily.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class ClientViewSource` `  Inherits System.Windows.DependencyObject`
C#
`public class ClientViewSource : System.Windows.DependencyObject`

Inheritance Hierarchy

System.Object
  System.Windows.Threading.DispatcherObject
    System.Windows.DependencyObject
      C1.Data.DataSource.ClientViewSource
        C1.Data.Entities.EntityViewSource
        C1.Silverlight.Data.RiaServices.RiaViewSource

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

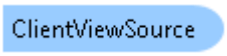
See Also

Reference

ClientViewSource Members
C1.Data.DataSource Namespace

Overview
Data source object exposing data from C1.Data.ClientCacheBase to which GUI controls can bind. Using a ClientViewSource, you can load, filter, group, and sort data easily.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class ClientViewSource` `  Inherits System.Windows.DependencyObject`
C#
`public class ClientViewSource : System.Windows.DependencyObject`

Inheritance Hierarchy

System.Object
  System.Windows.Threading.DispatcherObject

System.Windows.DependencyObject
C1.Data.DataSource.ClientViewSource
C1.Data.Entities.EntityViewSource
C1.Silverlight.Data.RiaServices.RiaViewSource

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Members
C1.Data.DataSource Namespace

Members
Properties Methods Events

The following tables list the members exposed by ClientViewSource.

Public Constructors

	Name	Description
	ClientViewSource Constructor	Initializes a new instance of the ClientViewSource class.

Top

Public Properties

	Name	Description
	AutoLoad	Gets or sets a value indicating whether Load is automatically invoked on startup and when a change occurs that impacts the query composed by the ClientViewSource. The default is True.
	BaseView	Gets or sets an instance of C1.Data.ClientView<T> that the ClientViewSource uses as the base for composing queries.

	CacheTimeout	Gets or sets the period of time entities loaded in virtual mode are kept in the cache without checking whether they are needed or not. If an entity was neither used nor considered needed for a period of time longer than CacheTimeout, ClientViewSource may evict it from the cache.
	CurrentClientView	Gets the current client view used to load entities, or null in virtual mode.
	DataView	Gets the current view of entities resulting from the last load operation.
	DependencyObjectType	(Inherited from System.Windows.DependencyObject)
	Dispatcher	(Inherited from System.Windows.Threading.DispatcherObject)
	FilterDescriptors	Gets the collection of FilterDescriptor objects used when performing loads.
	FilterOperator	Gets or sets the logical operator used for combinining FilterDescriptors in the filter collection. The default value is FilterDescriptorLogicalOperator.And.
	GroupDescriptors	Gets the collection of GroupDescriptor objects used to organize the loaded entities into groups.
	Include	Gets or sets a comma-separated list of property paths that specify related objects to include during the Load operation.
	IsLoadingData	Gets a value indicating whether the ClientViewSource is currently loading data.
	IsSealed	(Inherited from System.Windows.DependencyObject)
	LoadCommand	Gets an System.Windows.Input.ICommand that invokes Load on this

		ClientViewSource.
	LoadDelay	Gets or sets the delay before an automatic data loading operation is started. It is the delay from the time a change prompting automatic load occurs until the time the resulting Load is started. The default delay is 25 milliseconds.
	LoadSize	Gets or sets the maximum number of items to load each time a Load is executed. When equal to 0, all requested entities will be loaded. The default is 0.
	MoveToFirstOnLoad	Gets or sets a value indicating that the first item must be made current after Load operation is completed if current item was not set by other means.
	Name	Gets a name of this ClientViewSource to reference it in a C1DataSource.ViewSources collection. By default it is determined by the EntitySetName (for Entity Framework) or QueryName (for RIA Services), but it can be overrided by NameOverride.
	NameOverride	Gets or sets a value that overrides the value of the Name property.
	PageSize	Gets or sets the number of items displayed on each page of the DataView, or the number of items to fetch in each query in virtual mode, or 0 to disable paging.
	SortDescriptors	Gets the collection of SortDescriptor objects used to sort the data.
	VirtualMode	Gets or sets a value indicating whether the ClientViewSource is in virtual mode. Virtual mode is an innovative technology allowing to bind GUI controls directly to very large data sets without delays and performance degradation and without inconvenience of paging. By default, virtual mode is disabled (the default value is VirtualModeKind.None).

Top

Public Methods

	Name	Description
	ClearValue	Overloaded. (Inherited from System.Windows.DependencyObject)
	CoerceValue	(Inherited from System.Windows.DependencyObject)
	DeferLoad	Used to group changes to multiple load-affecting properties together, deferring the resulting load operations so a single load operation is performed in the end, that is, when the object returned from this method is disposed.
	Equals	(Inherited from System.Windows.DependencyObject)
	GetHashCode	(Inherited from System.Windows.DependencyObject)
	GetLocalValueEnumerator	(Inherited from System.Windows.DependencyObject)
	GetValue	(Inherited from System.Windows.DependencyObject)
	InvalidateProperty	(Inherited from System.Windows.DependencyObject)
	Load	Starts a load operation. Any pending load will be implicitly canceled.
	LoadRange	If in virtual mode, loads a specific range of entities.
	ReadLocalValue	(Inherited from System.Windows.DependencyObject)
	Refresh	Starts a load operation ignoring the client-side cache. Any pending load will be implicitly canceled.
	SetCurrentValue	(Inherited from System.Windows.DependencyObject)

	SetValue	Overloaded. (Inherited from System.Windows.DependencyObject)

Top

Public Events

	Name	Description
	LoadedData	Occurs when a load operation is completed, or when an exception was thrown during the load operation.
	PropertyChanged	Occurs when a property value changes.

Top

See Also

Reference

ClientViewSource Class
C1.Data.DataSource Namespace

ClientViewSource Constructor
Initializes a new instance of the ClientViewSource class.

Syntax

Visual Basic (Declaration)
`Public Function New()`
C#
`public ClientViewSource()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

Methods
For a list of all members of this type, see ClientViewSource members.

Public Methods

	Name	Description
	ClearValue	Overloaded. (Inherited from System.Windows.DependencyObject)
	CoerceValue	(Inherited from System.Windows.DependencyObject)
	DeferLoad	Used to group changes to multiple load-affecting properties together, deferring the resulting load operations so a single load operation is performed in the end, that is, when the object returned from this method is disposed.
	Equals	(Inherited from System.Windows.DependencyObject)
	GetHashCode	(Inherited from System.Windows.DependencyObject)
	GetLocalValueEnumerator	(Inherited from System.Windows.DependencyObject)
	GetValue	(Inherited from System.Windows.DependencyObject)
	InvalidateProperty	(Inherited from System.Windows.DependencyObject)
	Load	Starts a load operation. Any pending load will be implicitly canceled.
	LoadRange	If in virtual mode, loads a specific range of entities.
	ReadLocalValue	(Inherited from System.Windows.DependencyObject)

	Refresh	Starts a load operation ignoring the client-side cache. Any pending load will be implicitly canceled.
	SetCurrentValue	(Inherited from System.Windows.DependencyObject)
	SetValue	Overloaded. (Inherited from System.Windows.DependencyObject)

Top

See Also

Reference

ClientViewSource Class
C1.Data.DataSource Namespace

DeferLoad Method

Used to group changes to multiple load-affecting properties together, deferring the resulting load operations so a single load operation is performed in the end, that is, when the object returned from this method is disposed.

Syntax

Visual Basic (Declaration)
`Public Function DeferLoad() As IDisposable`
C#
`public IDisposable DeferLoad()`

Return Value

An System.IDisposable object that will trigger a Load operation when disposed using the System.IDisposable.Dispose method.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

Load Method
Starts a load operation. Any pending load will be implicitly canceled.

Syntax

Visual Basic (Declaration)
`Public Sub Load()`
C#
`public void Load()`

Remarks

If the entities are already in the cache, there will be no roundtrip to the server.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

LoadRange Method
The index of the first item to load.

The number of entities to load.

If in virtual mode, loads a specific range of entities.

Syntax

Visual Basic (Declaration)

`Public Sub LoadRange( _`
`  ByVal start As Integer, _`
`  ByVal length As Integer _`
`)`

C#

`public void LoadRange(`
`  int start,`
`  int length`
`)`

Parameters

start

The index of the first item to load.

length

The number of entities to load.

Exceptions

Exception	Description
System.InvalidOperationException	VirtualMode is VirtualModeKind.None.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

Refresh Method

Starts a load operation ignoring the client-side cache. Any pending load will be implicitly canceled.

Syntax

Visual Basic (Declaration)
`Public Sub Refresh()`
C#
`public void Refresh()`

Remarks

Use this method to refresh data with any changes that may have occurred on the server

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

Properties

For a list of all members of this type, see ClientViewSource members.

Public Properties

	Name	Description
	AutoLoad	Gets or sets a value indicating whether Load is automatically invoked on startup and when a change occurs that impacts the query composed by the ClientViewSource. The default is True.
	BaseView	Gets or sets an instance of C1.Data.ClientView<T> that the

		ClientViewSource uses as the base for composing queries.
	CacheTimeout	Gets or sets the period of time entities loaded in virtual mode are kept in the cache without checking whether they are needed or not. If an entity was neither used nor considered needed for a period of time longer than CacheTimeout, ClientViewSource may evict it from the cache.
	CurrentClientView	Gets the current client view used to load entities, or null in virtual mode.
	DataView	Gets the current view of entities resulting from the last load operation.
	DependencyObjectType	(Inherited from System.Windows.DependencyObject)
	Dispatcher	(Inherited from System.Windows.Threading.DispatcherObject)
	FilterDescriptors	Gets the collection of FilterDescriptor objects used when performing loads.
	FilterOperator	Gets or sets the logical operator used for combinining FilterDescriptors in the filter collection. The default value is FilterDescriptorLogicalOperator.And.
	GroupDescriptors	Gets the collection of GroupDescriptor objects used to organize the loaded entities into groups.
	Include	Gets or sets a comma-separated list of property paths that specify related objects to include during the Load operation.
	IsLoadingData	Gets a value indicating whether the ClientViewSource is currently loading data.
	IsSealed	(Inherited from System.Windows.DependencyObject)

	LoadCommand	Gets an System.Windows.Input.ICommand that invokes Load on this ClientViewSource.
	LoadDelay	Gets or sets the delay before an automatic data loading operation is started. It is the delay from the time a change prompting automatic load occurs until the time the resulting Load is started. The default delay is 25 milliseconds.
	LoadSize	Gets or sets the maximum number of items to load each time a Load is executed. When equal to 0, all requested entities will be loaded. The default is 0.
	MoveToFirstOnLoad	Gets or sets a value indicating that the first item must be made current after Load operation is completed if current item was not set by other means.
	Name	Gets a name of this ClientViewSource to reference it in a C1DataSource.ViewSources collection. By default it is determined by the EntitySetName (for Entity Framework) or QueryName (for RIA Services), but it can be overrided by NameOverride.
	NameOverride	Gets or sets a value that overrides the value of the Name property.
	PageSize	Gets or sets the number of items displayed on each page of the DataView, or the number of items to fetch in each query in virtual mode, or 0 to disable paging.
	SortDescriptors	Gets the collection of SortDescriptor objects used to sort the data.
	VirtualMode	Gets or sets a value indicating whether the ClientViewSource is in virtual mode. Virtual mode is an innovative technology allowing to bind GUI controls directly to very large data sets without delays and performance degradation and without inconvenience of paging. By default, virtual mode is disabled (the default value is

		VirtualModeKind.None).

Top

See Also

Reference

ClientViewSource Class
C1.Data.DataSource Namespace

AutoLoad Property
Gets or sets a value indicating whether Load is automatically invoked on startup and when a change occurs that impacts the query composed by the ClientViewSource. The default is True.

Syntax

Visual Basic (Declaration)
`Public Property AutoLoad As Boolean`
C#
`public bool AutoLoad {get; set;}`

Remarks

When AutoLoad is True, any property change affecting the load query will automatically invoke a Load after the specified LoadDelay. Examples of properties that impact the query are PageSize and FilterOperator. Also, changes to dependency object collections like FilterDescriptors and changes to the dependency properties on elements contained in those collections will affect the query and prompt an automatic Load.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

BaseView Property
Gets or sets an instance of C1.Data.ClientView<T> that the ClientViewSource uses as the base for composing queries.

Syntax

Visual Basic (Declaration)
`Public Property BaseView As View`
C#
`public View BaseView {get; set;}`

Exceptions

Exception	Description
System.ArgumentException	The value is not null and not an instance of C1.Data.ClientView<T>.

Remarks

The ClientViewSource applies filtering, sorting, grouping, and paging to its BaseView.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

CacheTimeout Property
Gets or sets the period of time entities loaded in virtual mode are kept in the cache without checking whether they are needed or not. If an entity was neither used nor considered needed for a period of time longer than CacheTimeout, ClientViewSource may evict it from the cache.

Syntax

Visual Basic (Declaration)
`Public Property CacheTimeout As TimeSpan`
C#
`public TimeSpan CacheTimeout {get; set;}`

Remarks

This property is not used if VirtualMode is VirtualModeKind.None.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

CurrentClientView Property
Gets the current client view used to load entities, or null in virtual mode.

Syntax

Visual Basic (Declaration)
`Public Property CurrentClientView As View`
C#
`public View CurrentClientView {get; set;}`

Remarks

Using CurrentClientView, you can build client views on top of the ClientViewSource by applying live view operators to the CurrentClientView.

The value of this property changes and the PropertyChanged event is raised whenever the query used to load entities changes.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

DataView Property
Gets the current view of entities resulting from the last load operation.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property DataView As ClientCollectionView`
C#
`public ClientCollectionView DataView {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

FilterDescriptors Property
Gets the collection of FilterDescriptor objects used when performing loads.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property FilterDescriptors As FilterDescriptorCollection`
C#
`public FilterDescriptorCollection FilterDescriptors {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

FilterOperator Property
Gets or sets the logical operator used for combinining FilterDescriptors in the filter collection. The default value is FilterDescriptorLogicalOperator.And.

Syntax

Visual Basic (Declaration)
`Public Property FilterOperator As FilterDescriptorLogicalOperator`
C#
`public FilterDescriptorLogicalOperator FilterOperator {get; set;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

GroupDescriptors Property
Gets the collection of GroupDescriptor objects used to organize the loaded entities into groups.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property GroupDescriptors As GroupDescriptorCollection`
C#
`public GroupDescriptorCollection GroupDescriptors {get;}`

Remarks

Grouping only works in WPF and Silverlight. It is ignored in WinForms because WinForms data binding does not support grouping.

When a GroupDescriptor is applied, the data will inherently be sorted by the grouped property. To force a grouped property to be sorted in descending order, add a SortDescriptor to the SortDescriptors collection for that property using the Descending direction.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

Include Property
Gets or sets a comma-separated list of property paths that specify related objects to include during the Load operation.

Syntax

Visual Basic (Declaration)
`Public Property Include As String`
C#
`public string Include {get; set;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

IsLoadingData Property
Gets a value indicating whether the ClientViewSource is currently loading data.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property IsLoadingData As Boolean`
C#
`public bool IsLoadingData {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

LoadCommand Property
Gets an System.Windows.Input.ICommand that invokes Load on this ClientViewSource.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property LoadCommand As ICommand`
C#
`public ICommand LoadCommand {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

LoadDelay Property
Gets or sets the delay before an automatic data loading operation is started. It is the delay from the time a change prompting automatic load occurs until the time the resulting Load is started. The default delay is 25 milliseconds.

Syntax

Visual Basic (Declaration)
`Public Property LoadDelay As TimeSpan`
C#
`public TimeSpan LoadDelay {get; set;}`

Remarks

Multiple changes that occur within the specified time span are aggregated into a single Load operation. For every change that occurs, the delay timer is reset. This allows many changes to be combined into a single call as long as each change occurs within the specified delay from the last. Once the delay timer is allowed to elapse without a change occurring, Load will be invoked.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

LoadSize Property
Gets or sets the maximum number of items to load each time a Load is executed. When equal to 0, all requested entities will be loaded. The default is 0.

Syntax

Visual Basic (Declaration)
`Public Property LoadSize As Integer`
C#
`public int LoadSize {get; set;}`

Remarks

When PageSize and LoadSize are both non-zero, entities will be loaded using the multiple of PageSize nearest LoadSize. This allows multiple pages to be loaded at once without loading partial pages.

This property is ignored when the ClientViewSource is in virtual mode.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

MoveToFirstOnLoad Property
Gets or sets a value indicating that the first item must be made current after Load operation is completed if current item was not set by other means.

Syntax

Visual Basic (Declaration)
`Public Property MoveToFirstOnLoad As Boolean`
C#
`public bool MoveToFirstOnLoad {get; set;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

Name Property
Gets a name of this ClientViewSource to reference it in a C1DataSource.ViewSources collection. By default it is determined by the EntitySetName (for Entity Framework) or QueryName (for RIA Services), but it can be overrided by NameOverride.

Syntax

Visual Basic (Declaration)
`Public Overridable ReadOnly Property Name As String`
C#
`public virtual string Name {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

NameOverride Property
Gets or sets a value that overrides the value of the Name property.

Syntax

Visual Basic (Declaration)
`Public Property NameOverride As String`
C#
`public string NameOverride {get; set;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

PageSize Property
Gets or sets the number of items displayed on each page of the DataView, or the number of items to fetch in each query in virtual mode, or 0 to disable paging.

Syntax

Visual Basic (Declaration)
`Public Property PageSize As Integer`
C#
`public int PageSize {get; set;}`

Remarks

If not in the virtual mode, a non-zero page size will cause the number of entities loaded with each Load operation to be limited, using server-side paging.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

SortDescriptors Property
Gets the collection of SortDescriptor objects used to sort the data.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property SortDescriptors As SortDescriptorCollection`

C#

```
public SortDescriptorCollection SortDescriptors {get;}
```

Remarks

In a Load operation, the SortDescriptors are used to perform server-side sorting. The specified sorting is also applied on the client side when changes are made on the client to the loaded entities, with the DataView reflecting the changes.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

VirtualMode Property
Gets or sets a value indicating whether the ClientViewSource is in virtual mode. Virtual mode is an innovative technology allowing to bind GUI controls directly to very large data sets without delays and performance degradation and without inconvenience of paging. By default, virtual mode is disabled (the default value is VirtualModeKind.None).

Syntax

Visual Basic (Declaration)

```
Public Property VirtualMode As VirtualModeKind
```

C#

```
public VirtualModeKind VirtualMode {get; set;}
```

Remarks

ClientViewSource in virtual mode intelligently and transparently both for the end user and for the developer loads only the entities that need to be displayed on the screen.

To enable virtual mode, set this property to Managed if you have a control handler with VirtualMode set to True; otherwise, set it to VirtualModeKind.Unmanaged.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members
VirtualModeKind Enumeration

Events
>

Name	Description
LoadedData	Occurs when a load operation is completed, or when an exception was thrown during the load operation.
PropertyChanged	Occurs when a property value changes.

Top

See Also

Reference

ClientViewSource Class
C1.Data.DataSource Namespace

LoadedData Event
Occurs when a load operation is completed, or when an exception was thrown during the load operation.

Syntax

Visual Basic (Declaration)	

```
Public Event LoadedData As EventHandler(Of ClientViewLoadedEventArgs)
```

C#

```
public event EventHandler<ClientViewLoadedEventArgs> LoadedData
```

Event Data

The event handler receives an argument of type ClientViewLoadedEventArgs containing data related to this event. The following **ClientViewLoadedEventArgs** properties provide information specific to this event.

Property	Description
Error	Gets the loading error if the loading failed.
HasError	Gets a value indicating whether the loading has failed. If true, inspect the Error property for details.
IsErrorHandled	Gets a value indicating whether the loading error has been marked as handled by calling MarkErrorAsHandled.
Items	Gets all entities loaded by a client view.
TotalItemCount	Gets the total number of rows for the original query without any paging applied to it.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

PropertyChanged Event
Occurs when a property value changes.

Syntax

Visual Basic (Declaration)	
`Public Event PropertyChanged As PropertyChangedEventHandler`	
C#	
`public event PropertyChangedEventHandler PropertyChanged`	

Event Data

The event handler receives an argument of type PropertyChangedEventArgs containing data related to this event. The following **PropertyChangedEventArgs** properties provide information specific to this event.

Property	Description
PropertyName	

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

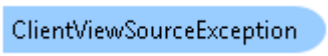
See Also

Reference

ClientViewSource Class
ClientViewSource Members

ClientViewSourceException
This exception indicates that a ClientViewSource is miconfigured or an error has occurred during the ClientViewSource.Load operation.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class ClientViewSourceException` `  Inherits System.Exception`
C#
`public class ClientViewSourceException : System.Exception`

Inheritance Hierarchy

System.Object
  System.Exception
    C1.Data.DataSource.ClientViewSourceException

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

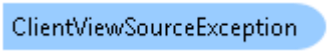
See Also

Reference

ClientViewSourceException Members
C1.Data.DataSource Namespace

Overview
This exception indicates that a ClientViewSource is miconfigured or an error has occurred during the ClientViewSource.Load operation.

Object Model



Syntax

Visual Basic (Declaration)

```
Public Class ClientViewSourceException
   Inherits System.Exception
```

C#

```
public class ClientViewSourceException : System.Exception
```

Inheritance Hierarchy

System.Object
  System.Exception
    C1.Data.DataSource.ClientViewSourceException

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSourceException Members
C1.Data.DataSource Namespace

Members
Properties Methods

The following tables list the members exposed by ClientViewSourceException.

Public Constructors

	Name	Description
	ClientViewSourceException Constructor	Overloaded.

Top

Public Properties

	Name	Description

	Data	(Inherited from System.Exception)
	HelpLink	(Inherited from System.Exception)
	InnerException	(Inherited from System.Exception)
	Message	(Inherited from System.Exception)
	Source	(Inherited from System.Exception)
	StackTrace	(Inherited from System.Exception)
	TargetSite	(Inherited from System.Exception)

Top

Public Methods

	Name	Description
	GetBaseException	(Inherited from System.Exception)
	GetObjectData	(Inherited from System.Exception)
	GetType	(Inherited from System.Exception)
	ToString	(Inherited from System.Exception)

Top

See Also

Reference

ClientViewSourceException Class
C1.Data.DataSource Namespace

ClientViewSourceException Constructor

Overload List

Overload	Description
ClientViewSourceException Constructor()	Initializes a new instance of the ClientViewSourceException class.
ClientViewSourceException Constructor(String)	Initializes a new instance of the ClientViewSourceException class with a specified error message.
ClientViewSourceException Constructor(String,Exception)	Initializes a new instance of the ClientViewSourceException class with a specified error message, and a reference to the inner exception that is the cause of this exception.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSourceException Class
ClientViewSourceException Members

ClientViewSourceException Constructor()
Initializes a new instance of the ClientViewSourceException class.

Syntax

Visual Basic (Declaration)
`Public Function New()`
C#
`public ClientViewSourceException()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSourceException Class
ClientViewSourceException Members
Overload List

ClientViewSourceException Constructor(String)
The error message that explains the reason for the exception.

Initializes a new instance of the ClientViewSourceException class with a specified error message.

Syntax

Visual Basic (Declaration)

`Public Function New( _`
`  ByVal message As String _`
`)`

C#

`public ClientViewSourceException(`
`  string message`
`)`

Parameters

message

The error message that explains the reason for the exception.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSourceException Class
ClientViewSourceException Members
Overload List

ClientViewSourceException Constructor(String,Exception)
The error message that explains the reason for the exception.

The exception that is the cause of the current exception.

Initializes a new instance of the ClientViewSourceException class with a specified error message, and a reference to the inner exception that is the cause of this exception.

Syntax

Visual Basic (Declaration)

```
Public Function New( _
   ByVal message As String, _
   ByVal inner As Exception _
)
```

C#

```
public ClientViewSourceException(
   string message,
   Exception inner
)
```

Parameters

message

The error message that explains the reason for the exception.

inner

The exception that is the cause of the current exception.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

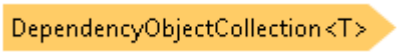
Reference

ClientViewSourceException Class
ClientViewSourceException Members
Overload List

DependencyObjectCollection<T>
The type of objects in the collection. Must be derived from System.Windows.DependencyObject.

An observable collection of dependency objects.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class DependencyObjectCollection(Of T As DependencyObject)` `  Inherits System.Collections.ObjectModel.ObservableCollection(Of T)`
C#

`public class DependencyObjectCollection<T> :`
`System.Collections.ObjectModel.ObservableCollection<T>`
`where T: DependencyObject`

Type Parameters

T

The type of objects in the collection. Must be derived from System.Windows.DependencyObject.

Inheritance Hierarchy

System.Object
System.Collections.ObjectModel.Collection<T>
System.Collections.ObjectModel.ObservableCollection<T>
C1.Data.DataSource.DependencyObjectCollection<T>
C1.Data.DataSource.FilterDescriptorCollection
C1.Data.DataSource.GroupDescriptorCollection
C1.Data.DataSource.SortDescriptorCollection

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

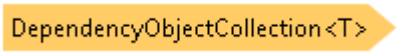
Reference

DependencyObjectCollection<T> Members
C1.Data.DataSource Namespace

Overview
The type of objects in the collection. Must be derived from System.Windows.DependencyObject.

An observable collection of dependency objects.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class DependencyObjectCollection(Of T As DependencyObject)` `  Inherits System.Collections.ObjectModel.ObservableCollection(Of T)`
C#

`public class DependencyObjectCollection<T> :`
`System.Collections.ObjectModel.ObservableCollection<T>`
`where T: DependencyObject`

Type Parameters

T

The type of objects in the collection. Must be derived from System.Windows.DependencyObject.

Inheritance Hierarchy

System.Object
  System.Collections.ObjectModel.Collection<T>
    System.Collections.ObjectModel.ObservableCollection<T>
      C1.Data.DataSource.DependencyObjectCollection<T>
        C1.Data.DataSource.FilterDescriptorCollection
        C1.Data.DataSource.GroupDescriptorCollection
        C1.Data.DataSource.SortDescriptorCollection

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

DependencyObjectCollection<T> Members
C1.Data.DataSource Namespace

Members
Properties Methods Events

The following tables list the members exposed by DependencyObjectCollection<T>.

Public Constructors

	Name	Description
	DependencyObjectCollection<T> Constructor	

Top

Public Properties

	Name	Description
	Count	(Inherited from System.Collections.ObjectModel.Collection<T>)
	Item	(Inherited from System.Collections.ObjectModel.Collection<T>)

Top

Public Methods

	Name	Description
	Add	(Inherited from System.Collections.ObjectModel.Collection<T>)
	Clear	(Inherited from System.Collections.ObjectModel.Collection<T>)
	Contains	(Inherited from System.Collections.ObjectModel.Collection<T>)
	CopyTo	(Inherited from System.Collections.ObjectModel.Collection<T>)
	GetEnumerator	(Inherited from System.Collections.ObjectModel.Collection<T>)
	IndexOf	(Inherited from System.Collections.ObjectModel.Collection<T>)
	Insert	(Inherited from System.Collections.ObjectModel.Collection<T>)
	Move	(Inherited from System.Collections.ObjectModel.ObservableCollection<T>)
	Remove	(Inherited from System.Collections.ObjectModel.Collection<T>)
	RemoveAt	(Inherited from System.Collections.ObjectModel.Collection<T>)

Top

Public Events

	Name	Description
	CollectionChanged	(Inherited from System.Collections.ObjectModel.ObservableCollection<T>)

Top

See Also

Reference

DependencyObjectCollection<T> Class
C1.Data.DataSource Namespace

DependencyObjectCollection<T> Constructor

Syntax

Visual Basic (Declaration)
`Public Function New()`
C#
`public DependencyObjectCollection<T>()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

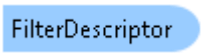
See Also

Reference

DependencyObjectCollection<T> Class
DependencyObjectCollection<T> Members

FilterDescriptor
Descriptor used by the ClientViewSource to filter data in queries.

Object Model



Syntax

Visual Basic (Declaration)	
`Public Class FilterDescriptor` `  Inherits System.Windows.DependencyObject`	
C#	
`public class FilterDescriptor : System.Windows.DependencyObject`	

Inheritance Hierarchy

System.Object
  System.Windows.Threading.DispatcherObject
    System.Windows.DependencyObject
      C1.Data.DataSource.FilterDescriptor

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

FilterDescriptor Members
C1.Data.DataSource Namespace

Overview
Descriptor used by the ClientViewSource to filter data in queries.

Object Model

Syntax

Visual Basic (Declaration)	

```
Public Class FilterDescriptor
   Inherits System.Windows.DependencyObject
```

C#

```
public class FilterDescriptor : System.Windows.DependencyObject
```

Inheritance Hierarchy

System.Object
  System.Windows.Threading.DispatcherObject
    System.Windows.DependencyObject
      C1.Data.DataSource.FilterDescriptor

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

FilterDescriptor Members
C1.Data.DataSource Namespace

Members
Fields Properties Methods

The following tables list the members exposed by FilterDescriptor.

Public Constructors

	Name	Description
	FilterDescriptor Constructor	Overloaded.

Top

Public Fields

	Name	Description

S	DefaultIgnoredValue	The default value of the IgnoredValue property.
S	IgnoredValueProperty	The System.Windows.DependencyProperty for the IgnoredValue property.
S	IsCaseSensitiveProperty	The DependencyProperty for the IsCaseSensitive property.
S	OperatorProperty	The DependencyProperty for the Operator property.
S	PropertyPathProperty	The DependencyProperty for the PropertyPath property.
S	ValueProperty	The DependencyProperty for the Value property.

Top

Public Properties

	Name	Description
	DependencyObjectType	(Inherited from System.Windows.DependencyObject)
	Dispatcher	(Inherited from System.Windows.Threading.DispatcherObject)
	IgnoredValue	Gets or sets the value for the right operand for which this filter should be ignored.
	IsCaseSensitive	Gets or sets a value indicating whether the FilterDescriptor is case sensitive for string values.
	IsSealed	(Inherited from System.Windows.DependencyObject)
	Operator	Gets or sets the filter operator.
	PropertyPath	Gets or sets the name of the property path used as the left operand.
	Value	Gets or sets the value of the right operand.

Top

Public Methods

	Name	Description
	ClearValue	Overloaded. (Inherited from System.Windows.DependencyObject)
	CoerceValue	(Inherited from System.Windows.DependencyObject)
	Equals	(Inherited from System.Windows.DependencyObject)
	GetHashCode	(Inherited from System.Windows.DependencyObject)
	GetLocalValueEnumerator	(Inherited from System.Windows.DependencyObject)
	GetValue	(Inherited from System.Windows.DependencyObject)
	InvalidateProperty	(Inherited from System.Windows.DependencyObject)
	ReadLocalValue	(Inherited from System.Windows.DependencyObject)
	SetCurrentValue	(Inherited from System.Windows.DependencyObject)
	SetValue	Overloaded. (Inherited from System.Windows.DependencyObject)

Top

See Also

Reference

FilterDescriptor Class
C1.Data.DataSource Namespace

FilterDescriptor Constructor

Overload List

Overload	Description
FilterDescriptor Constructor(String,FilterOperator,Object)	Initializes a new instance of the FilterDescriptor class with the specified property to use for filtering, the operator to use when evaluating the filtering check, and the filter value.
FilterDescriptor Constructor()	Initializes a new instance of the FilterDescriptor class with default values.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

FilterDescriptor Class
FilterDescriptor Members

FilterDescriptor Constructor(String,FilterOperator,Object)
The property path to use for filtering.

The kind of comparison to use.

The value to use when filtering.

Initializes a new instance of the FilterDescriptor class with the specified property to use for filtering, the operator to use when evaluating the filtering check, and the filter value.

Syntax

Visual Basic (Declaration)

```
Public Function New( _
   ByVal propertyPath As String, _
   ByVal filterOperator As FilterOperator, _
```

```
   ByVal filterValue As Object _
)
```

C#

```
public FilterDescriptor(
   string propertyPath,
   FilterOperator filterOperator,
   object filterValue
)
```

Parameters

propertyPath

The property path to use for filtering.

filterOperator

The kind of comparison to use.

filterValue

The value to use when filtering.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

FilterDescriptor Class
FilterDescriptor Members
Overload List

FilterDescriptor Constructor()
Initializes a new instance of the FilterDescriptor class with default values.

Syntax

Visual Basic (Declaration)

```
Public Function New()
```

C#

```
public FilterDescriptor()
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

FilterDescriptor Class
FilterDescriptor Members
Overload List

Properties
>

Name	Description
DependencyObjectType	(Inherited from System.Windows.DependencyObject)
Dispatcher	(Inherited from System.Windows.Threading.DispatcherObject)
IgnoredValue	Gets or sets the value for the right operand for which this filter should be ignored.
IsCaseSensitive	Gets or sets a value indicating whether the FilterDescriptor is case sensitive for string values.
IsSealed	(Inherited from System.Windows.DependencyObject)
Operator	Gets or sets the filter operator.
PropertyPath	Gets or sets the name of the property path used as the left operand.
Value	Gets or sets the value of the right operand.

Top

See Also

Reference

FilterDescriptor Class
C1.Data.DataSource Namespace

IgnoredValue Property
Gets or sets the value for the right operand for which this filter should be ignored.

Syntax

Visual Basic (Declaration)
`Public Property IgnoredValue As Object`
C#
`public object IgnoredValue {get; set;}`

Remarks

If Value matches IgnoredValue, this filter will not be applied to the load query by the ClientViewSource. The IgnoredValue is compared to Value twice in the ClientViewSource. First, it is strictly compared using an System.Object.Equals(System.Object,System.Object) comparison. Second, both values are converted to type of the property specified by the PropertyPath and compared again. If either conversion matches, this filter is ignored.

For example, the following Value/IgnoredValue pairs will all match for an integer property and result in the filter being ignored: 0/0, 0/"0", "0"/"0", and "0"/0.

This property is set to DefaultIgnoredValue by default. The default value will only match if Value is also set to DefaultIgnoredValue.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

FilterDescriptor Class
FilterDescriptor Members

IsCaseSensitive Property
Gets or sets a value indicating whether the FilterDescriptor is case sensitive for string values.

Syntax

Visual Basic (Declaration)
`Public Property IsCaseSensitive As Boolean`
C#
`public bool IsCaseSensitive {get; set;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

FilterDescriptor Class
FilterDescriptor Members

Operator Property
Gets or sets the filter operator.

Syntax

Visual Basic (Declaration)
`Public Property Operator As FilterOperator`
C#
`public FilterOperator Operator {get; set;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

FilterDescriptor Class
FilterDescriptor Members

PropertyPath Property
Gets or sets the name of the property path used as the left operand.

Syntax

Visual Basic (Declaration)
`Public Property PropertyPath As String`
C#
`public string PropertyPath {get; set;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

FilterDescriptor Class
FilterDescriptor Members

Value Property
Gets or sets the value of the right operand.

Syntax

Visual Basic (Declaration)

```
Public Property Value As Object
```

C#

```
public object Value {get; set;}
```

Remarks

This will be used by the ClientViewSource to compose a filter for the load query. It will be applied following the pattern `[Entity].[PropertyPath] [Operator] [Value]`. For example, a query might look like `Customer.Name == "CurrentCustomerName"`.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

FilterDescriptor Class
FilterDescriptor Members

Fields
>

Name	Description
S DefaultIgnoredValue	The default value of the IgnoredValue property.
S IgnoredValueProperty	The System.Windows.DependencyProperty for the IgnoredValue property.
S IsCaseSensitiveProperty	The DependencyProperty for the IsCaseSensitive property.
S OperatorProperty	The DependencyProperty for the Operator property.
S PropertyPathProperty	The DependencyProperty for the PropertyPath property.
S ValueProperty	The DependencyProperty for the Value property.

Top

See Also

Reference

FilterDescriptor Class
C1.Data.DataSource Namespace

DefaultIgnoredValue Field
The default value of the IgnoredValue property.

Syntax

Visual Basic (Declaration)
`Public Shared ReadOnly DefaultIgnoredValue As Object`
C#
`public static readonly object DefaultIgnoredValue`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

FilterDescriptor Class
FilterDescriptor Members

IgnoredValueProperty Field
The System.Windows.DependencyProperty for the IgnoredValue property.

Syntax

Visual Basic (Declaration)
`Public Shared ReadOnly IgnoredValueProperty As DependencyProperty`
C#

```
public static readonly DependencyProperty IgnoredValueProperty
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

FilterDescriptor Class
FilterDescriptor Members

IsCaseSensitiveProperty Field
The DependencyProperty for the IsCaseSensitive property.

Syntax

Visual Basic (Declaration)
`Public Shared ReadOnly IsCaseSensitiveProperty As DependencyProperty`
C#
`public static readonly DependencyProperty IsCaseSensitiveProperty`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

FilterDescriptor Class
FilterDescriptor Members

OperatorProperty Field
The DependencyProperty for the Operator property.

Syntax

Visual Basic (Declaration)
`Public Shared ReadOnly OperatorProperty As DependencyProperty`
C#
`public static readonly DependencyProperty OperatorProperty`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

FilterDescriptor Class
FilterDescriptor Members

PropertyPathProperty Field
The DependencyProperty for the PropertyPath property.

Syntax

Visual Basic (Declaration)
`Public Shared ReadOnly PropertyPathProperty As DependencyProperty`
C#
`public static readonly DependencyProperty PropertyPathProperty`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

FilterDescriptor Class
FilterDescriptor Members

ValueProperty Field
The DependencyProperty for the Value property.

Syntax

Visual Basic (Declaration)
`Public Shared ReadOnly ValueProperty As DependencyProperty`
C#
`public static readonly DependencyProperty ValueProperty`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

FilterDescriptor Class
FilterDescriptor Members

FilterDescriptorCollection
Collection of FilterDescriptor dependency objects.

Object Model

Syntax

Visual Basic (Declaration)
`Public Class FilterDescriptorCollection` `  Inherits C1.Data.DataSource.DependencyObjectCollection(Of FilterDescriptor)`

C#

```
public class FilterDescriptorCollection :
C1.Data.DataSource.DependencyObjectCollection<FilterDescriptor>
```

Inheritance Hierarchy

System.Object
  System.Collections.ObjectModel.Collection<T>
    System.Collections.ObjectModel.ObservableCollection<T>
      C1.Data.DataSource.DependencyObjectCollection<T>
        C1.Data.DataSource.FilterDescriptorCollection

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

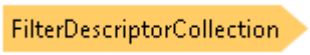
See Also

Reference

FilterDescriptorCollection Members
C1.Data.DataSource Namespace

Overview
Collection of FilterDescriptor dependency objects.

Object Model



Syntax

Visual Basic (Declaration)

```
Public Class FilterDescriptorCollection
   Inherits C1.Data.DataSource.DependencyObjectCollection(Of FilterDescriptor)
```

C#

```
public class FilterDescriptorCollection :
C1.Data.DataSource.DependencyObjectCollection<FilterDescriptor>
```

Inheritance Hierarchy

System.Object
  System.Collections.ObjectModel.Collection<T>
    System.Collections.ObjectModel.ObservableCollection<T>
      C1.Data.DataSource.DependencyObjectCollection<T>
        C1.Data.DataSource.FilterDescriptorCollection

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

FilterDescriptorCollection Members
C1.Data.DataSource Namespace

Members
Properties Methods Events

The following tables list the members exposed by FilterDescriptorCollection.

Public Constructors

	Name	Description
	FilterDescriptorCollection Constructor	Initializes a new instance of the FilterDescriptorCollection class.

Top

Public Properties

	Name	Description

	Count	(Inherited from System.Collections.ObjectModel.Collection<FilterDescriptor>)
	Item	(Inherited from System.Collections.ObjectModel.Collection<FilterDescriptor>)

Top

Public Methods

	Name	Description
	Add	(Inherited from System.Collections.ObjectModel.Collection<FilterDescriptor>)
	Clear	(Inherited from System.Collections.ObjectModel.Collection<FilterDescriptor>)
	Contains	(Inherited from System.Collections.ObjectModel.Collection<FilterDescriptor>)
	CopyTo	(Inherited from System.Collections.ObjectModel.Collection<FilterDescriptor>)
	GetEnumerator	(Inherited from System.Collections.ObjectModel.Collection<FilterDescriptor>)
	IndexOf	(Inherited from System.Collections.ObjectModel.Collection<FilterDescriptor>)
	Insert	(Inherited from System.Collections.ObjectModel.Collection<FilterDescriptor>)
	Move	(Inherited from System.Collections.ObjectModel.ObservableCollection<FilterDescriptor>)

	Remove	(Inherited from System.Collections.ObjectModel.Collection<FilterDescriptor>)
	RemoveAt	(Inherited from System.Collections.ObjectModel.Collection<FilterDescriptor>)

Top

Public Events

	Name	Description
	CollectionChanged	(Inherited from System.Collections.ObjectModel.ObservableCollection<FilterDescriptor>)

Top

See Also

Reference

FilterDescriptorCollection Class
C1.Data.DataSource Namespace

FilterDescriptorCollection Constructor
Initializes a new instance of the FilterDescriptorCollection class.

Syntax

Visual Basic (Declaration)
`Public Function New()`
C#
`public FilterDescriptorCollection()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

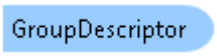
See Also

Reference

FilterDescriptorCollection Class
FilterDescriptorCollection Members

GroupDescriptor
Descriptor used by the ClientViewSource to group data returned from server-side queries.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class GroupDescriptor` `  Inherits System.Windows.DependencyObject`
C#
`public class GroupDescriptor : System.Windows.DependencyObject`

Inheritance Hierarchy

System.Object
  System.Windows.Threading.DispatcherObject
    System.Windows.DependencyObject
      C1.Data.DataSource.GroupDescriptor

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

GroupDescriptor Members
C1.Data.DataSource Namespace

Overview

Descriptor used by the ClientViewSource to group data returned from server-side queries.

Object Model

Syntax

Visual Basic (Declaration)
`Public Class GroupDescriptor` `  Inherits System.Windows.DependencyObject`
C#
`public class GroupDescriptor : System.Windows.DependencyObject`

Inheritance Hierarchy

System.Object
  System.Windows.Threading.DispatcherObject
    System.Windows.DependencyObject
      C1.Data.DataSource.GroupDescriptor

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

GroupDescriptor Members
C1.Data.DataSource Namespace

Members

Fields Properties Methods

The following tables list the members exposed by GroupDescriptor.

Public Constructors

	Name	Description
	GroupDescriptor Constructor	Overloaded.

Top

Public Fields

	Name	Description
S	PropertyPathProperty	The DependencyProperty for the PropertyPath property.

Top

Public Properties

	Name	Description
	DependencyObjectType	(Inherited from System.Windows.DependencyObject)
	Dispatcher	(Inherited from System.Windows.Threading.DispatcherObject)
	IsSealed	(Inherited from System.Windows.DependencyObject)
	PropertyPath	Gets or sets the name of the property path used to group data.

Top

Public Methods

	Name	Description
	ClearValue	Overloaded. (Inherited from System.Windows.DependencyObject)

	CoerceValue	(Inherited from System.Windows.DependencyObject)
	Equals	(Inherited from System.Windows.DependencyObject)
	GetHashCode	(Inherited from System.Windows.DependencyObject)
	GetLocalValueEnumerator	(Inherited from System.Windows.DependencyObject)
	GetValue	(Inherited from System.Windows.DependencyObject)
	InvalidateProperty	(Inherited from System.Windows.DependencyObject)
	ReadLocalValue	(Inherited from System.Windows.DependencyObject)
	SetCurrentValue	(Inherited from System.Windows.DependencyObject)
	SetValue	Overloaded. (Inherited from System.Windows.DependencyObject)

Top

See Also

Reference

GroupDescriptor Class
C1.Data.DataSource Namespace

GroupDescriptor Constructor

Overload List

Overload	Description
GroupDescriptor Constructor()	Initializes a new instance of the GroupDescriptor class.
GroupDescriptor Constructor(String)	Initializes a new instance of the GroupDescriptor class.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

GroupDescriptor Class
GroupDescriptor Members

GroupDescriptor Constructor()
Initializes a new instance of the GroupDescriptor class.

Syntax

Visual Basic (Declaration)	
`Public Function New()`	
C#	
`public GroupDescriptor()`	

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

GroupDescriptor Class
GroupDescriptor Members
Overload List

GroupDescriptor Constructor(String)
The group property path

Initializes a new instance of the GroupDescriptor class.

Syntax

Visual Basic (Declaration)

```
Public Function New( _
   ByVal propertyPath As String _
)
```

C#

```
public GroupDescriptor(
   string propertyPath
)
```

Parameters

propertyPath

The group property path

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

GroupDescriptor Class
GroupDescriptor Members
Overload List

Properties
>

Name	Description
DependencyObjectType	(Inherited from System.Windows.DependencyObject)
Dispatcher	(Inherited from System.Windows.Threading.DispatcherObject)
IsSealed	(Inherited from System.Windows.DependencyObject)
PropertyPath	Gets or sets the name of the property path used to group data.

Top

See Also

Reference

GroupDescriptor Class
C1.Data.DataSource Namespace

PropertyPath Property
Gets or sets the name of the property path used to group data.

Syntax

Visual Basic (Declaration)
`Public Property PropertyPath As String`
C#
`public string PropertyPath {get; set;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

GroupDescriptor Class
GroupDescriptor Members

Fields
For a list of all members of this type, see GroupDescriptor members.

Public Fields

	Name	Description
S	PropertyPathProperty	The DependencyProperty for the PropertyPath property.

Top

See Also

Reference

GroupDescriptor Class
C1.Data.DataSource Namespace

PropertyPathProperty Field
The DependencyProperty for the PropertyPath property.

Syntax

Visual Basic (Declaration)
`Public Shared ReadOnly PropertyPathProperty As DependencyProperty`
C#
`public static readonly DependencyProperty PropertyPathProperty`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

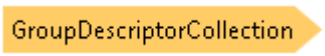
See Also

Reference

GroupDescriptor Class
GroupDescriptor Members

GroupDescriptorCollection
Collection of GroupDescriptor dependency objects.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class GroupDescriptorCollection` `  Inherits C1.Data.DataSource.DependencyObjectCollection(Of GroupDescriptor)`
C#
`public class GroupDescriptorCollection :` `C1.Data.DataSource.DependencyObjectCollection<GroupDescriptor>`

Inheritance Hierarchy

System.Object
  System.Collections.ObjectModel.Collection<T>
    System.Collections.ObjectModel.ObservableCollection<T>
      C1.Data.DataSource.DependencyObjectCollection<T>
        C1.Data.DataSource.GroupDescriptorCollection

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

GroupDescriptorCollection Members
C1.Data.DataSource Namespace

Overview
Collection of GroupDescriptor dependency objects.

Object Model

Syntax

Visual Basic (Declaration)

```
Public Class GroupDescriptorCollection
   Inherits C1.Data.DataSource.DependencyObjectCollection(Of GroupDescriptor)
```

C#

```
public class GroupDescriptorCollection :
C1.Data.DataSource.DependencyObjectCollection<GroupDescriptor>
```

Inheritance Hierarchy

System.Object
  System.Collections.ObjectModel.Collection<T>
    System.Collections.ObjectModel.ObservableCollection<T>
      C1.Data.DataSource.DependencyObjectCollection<T>
        C1.Data.DataSource.GroupDescriptorCollection

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

GroupDescriptorCollection Members
C1.Data.DataSource Namespace

Members
Properties Methods Events

The following tables list the members exposed by GroupDescriptorCollection.

Public Constructors

	Name	Description
	GroupDescriptorCollection Constructor	Initializes a new instance of the GroupDescriptorCollection class.

Top

Public Properties

	Name	Description
	Count	(Inherited from System.Collections.ObjectModel.Collection<GroupDescriptor>)
	Item	(Inherited from System.Collections.ObjectModel.Collection<GroupDescriptor>)

Top

Public Methods

	Name	Description
	Add	(Inherited from System.Collections.ObjectModel.Collection<GroupDescriptor>)
	Clear	(Inherited from System.Collections.ObjectModel.Collection<GroupDescriptor>)
	Contains	(Inherited from System.Collections.ObjectModel.Collection<GroupDescriptor>)
	CopyTo	(Inherited from System.Collections.ObjectModel.Collection<GroupDescriptor>)
	GetEnumerator	(Inherited from System.Collections.ObjectModel.Collection<GroupDescriptor>)
	IndexOf	(Inherited from System.Collections.ObjectModel.Collection<GroupDescriptor>)
	Insert	(Inherited from System.Collections.ObjectModel.Collection<GroupDescriptor>)

	Move	(Inherited from System.Collections.ObjectModel.ObservableCollection<GroupDescriptor>)
	Remove	(Inherited from System.Collections.ObjectModel.Collection<GroupDescriptor>)
	RemoveAt	(Inherited from System.Collections.ObjectModel.Collection<GroupDescriptor>)

Top

Public Events

	Name	Description
	CollectionChanged	(Inherited from System.Collections.ObjectModel.ObservableCollection<GroupDescriptor>)

Top

See Also

Reference

GroupDescriptorCollection Class
C1.Data.DataSource Namespace

GroupDescriptorCollection Constructor
Initializes a new instance of the GroupDescriptorCollection class.

Syntax

Visual Basic (Declaration)
`Public Function New()`
C#
`public GroupDescriptorCollection()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

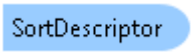
See Also

Reference

GroupDescriptorCollection Class
GroupDescriptorCollection Members

SortDescriptor
Descriptor used by the ClientViewSource to sort data returned from queries.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class SortDescriptor` `  Inherits System.Windows.DependencyObject`
C#
`public class SortDescriptor : System.Windows.DependencyObject`

Inheritance Hierarchy

System.Object
  System.Windows.Threading.DispatcherObject
    System.Windows.DependencyObject
      C1.Data.DataSource.SortDescriptor

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

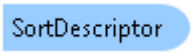
See Also

Reference

SortDescriptor Members
C1.Data.DataSource Namespace

Overview
Descriptor used by the ClientViewSource to sort data returned from queries.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class SortDescriptor` `  Inherits System.Windows.DependencyObject`
C#
`public class SortDescriptor : System.Windows.DependencyObject`

Inheritance Hierarchy

System.Object
  System.Windows.Threading.DispatcherObject
    System.Windows.DependencyObject
      C1.Data.DataSource.SortDescriptor

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SortDescriptor Members
C1.Data.DataSource Namespace

Members

Fields Properties Methods

The following tables list the members exposed by SortDescriptor.

Public Constructors

	Name	Description
	SortDescriptor Constructor	Overloaded.

Top

Public Fields

	Name	Description
S	DirectionProperty	The DependencyProperty for the Direction property.
S	PropertyPathProperty	The DependencyProperty for the PropertyPath property.

Top

Public Properties

	Name	Description
	DependencyObjectType	(Inherited from System.Windows.DependencyObject)
	Direction	Gets or sets the sort direction: Ascending or Descending.
	Dispatcher	(Inherited from System.Windows.Threading.DispatcherObject)
	IsSealed	(Inherited from System.Windows.DependencyObject)
	PropertyPath	Gets or sets the name of the property path used to sort data.

Top

Public Methods

	Name	Description
	ClearValue	Overloaded. (Inherited from System.Windows.DependencyObject)
	CoerceValue	(Inherited from System.Windows.DependencyObject)
	Equals	(Inherited from System.Windows.DependencyObject)
	GetHashCode	(Inherited from System.Windows.DependencyObject)
	GetLocalValueEnumerator	(Inherited from System.Windows.DependencyObject)
	GetValue	(Inherited from System.Windows.DependencyObject)
	InvalidateProperty	(Inherited from System.Windows.DependencyObject)
	ReadLocalValue	(Inherited from System.Windows.DependencyObject)
	SetCurrentValue	(Inherited from System.Windows.DependencyObject)
	SetValue	Overloaded. (Inherited from System.Windows.DependencyObject)

Top

See Also

Reference

SortDescriptor Class
C1.Data.DataSource Namespace

SortDescriptor Constructor

Overload List

Overload	Description
SortDescriptor Constructor()	Initializes a new instance of the SortDescriptor class.

SortDescriptor Constructor(String,ListSortDirection)	Initializes a new instance of the SortDescriptor class.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SortDescriptor Class
SortDescriptor Members

SortDescriptor Constructor()
Initializes a new instance of the SortDescriptor class.

Syntax

Visual Basic (Declaration)
`Public Function New()`
C#
`public SortDescriptor()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SortDescriptor Class
SortDescriptor Members
Overload List

SortDescriptor Constructor(String,ListSortDirection)

The sort property path

The sort direction

Initializes a new instance of the SortDescriptor class.

Syntax

Visual Basic (Declaration)

```
Public Function New( _
   ByVal propertyPath As String, _
   ByVal direction As ListSortDirection _
)
```

C#

```
public SortDescriptor(
   string propertyPath,
   ListSortDirection direction
)
```

Parameters

propertyPath

The sort property path

direction

The sort direction

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SortDescriptor Class
SortDescriptor Members
Overload List

Properties
>

Name	Description
DependencyObjectType	(Inherited from System.Windows.DependencyObject)
Direction	Gets or sets the sort direction: Ascending or Descending.
Dispatcher	(Inherited from System.Windows.Threading.DispatcherObject)
IsSealed	(Inherited from System.Windows.DependencyObject)
PropertyPath	Gets or sets the name of the property path used to sort data.

Top

See Also

Reference

SortDescriptor Class
C1.Data.DataSource Namespace

Direction Property
Gets or sets the sort direction: Ascending or Descending.

Syntax

Visual Basic (Declaration)

```
Public Property Direction As ListSortDirection
```

C#

```
public ListSortDirection Direction {get; set;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SortDescriptor Class
SortDescriptor Members

PropertyPath Property
Gets or sets the name of the property path used to sort data.

Syntax

Visual Basic (Declaration)
`Public Property PropertyPath As String`
C#
`public string PropertyPath {get; set;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SortDescriptor Class
SortDescriptor Members

Fields
For a list of all members of this type, see SortDescriptor members.

Public Fields

	Name	Description

S	DirectionProperty	The DependencyProperty for the Direction property.
S	PropertyPathProperty	The DependencyProperty for the PropertyPath property.

Top

See Also

Reference

SortDescriptor Class
C1.Data.DataSource Namespace

DirectionProperty Field
The DependencyProperty for the Direction property.

Syntax

Visual Basic (Declaration)
`Public Shared ReadOnly DirectionProperty As DependencyProperty`
C#
`public static readonly DependencyProperty DirectionProperty`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SortDescriptor Class
SortDescriptor Members

PropertyPathProperty Field
The DependencyProperty for the PropertyPath property.

Syntax

Visual Basic (Declaration)

```
Public Shared ReadOnly PropertyPathProperty As DependencyProperty
```

C#

```
public static readonly DependencyProperty PropertyPathProperty
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

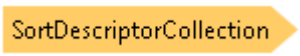
See Also

Reference

SortDescriptor Class
SortDescriptor Members

SortDescriptorCollection
Collection of SortDescriptor dependency objects.

Object Model



Syntax

Visual Basic (Declaration)

```
Public Class SortDescriptorCollection
   Inherits C1.Data.DataSource.DependencyObjectCollection(Of SortDescriptor)
```

C#

```
public class SortDescriptorCollection :
C1.Data.DataSource.DependencyObjectCollection<SortDescriptor>
```

Inheritance Hierarchy

System.Object
  System.Collections.ObjectModel.Collection<T>

System.Collections.ObjectModel.ObservableCollection<T>
C1.Data.DataSource.DependencyObjectCollection<T>
C1.Data.DataSource.SortDescriptorCollection

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

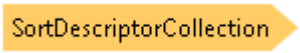
See Also

Reference

SortDescriptorCollection Members
C1.Data.DataSource Namespace

Overview
Collection of SortDescriptor dependency objects.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class SortDescriptorCollection` `  Inherits C1.Data.DataSource.DependencyObjectCollection(Of SortDescriptor)`
C#
`public class SortDescriptorCollection :` `C1.Data.DataSource.DependencyObjectCollection<SortDescriptor>`

Inheritance Hierarchy

System.Object
System.Collections.ObjectModel.Collection<T>
System.Collections.ObjectModel.ObservableCollection<T>
C1.Data.DataSource.DependencyObjectCollection<T>
C1.Data.DataSource.SortDescriptorCollection

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SortDescriptorCollection Members
C1.Data.DataSource Namespace

Members
Properties Methods Events

The following tables list the members exposed by SortDescriptorCollection.

Public Constructors

	Name	Description
	SortDescriptorCollection Constructor	Initializes a new instance of the SortDescriptorCollection class.

Top

Public Properties

	Name	Description
	Count	(Inherited from System.Collections.ObjectModel.Collection<SortDescriptor>)
	Item	(Inherited from System.Collections.ObjectModel.Collection<SortDescriptor>)

Top

Public Methods

	Name	Description

	Add	(Inherited from System.Collections.ObjectModel.Collection<SortDescriptor>)
	Clear	(Inherited from System.Collections.ObjectModel.Collection<SortDescriptor>)
	Contains	(Inherited from System.Collections.ObjectModel.Collection<SortDescriptor>)
	CopyTo	(Inherited from System.Collections.ObjectModel.Collection<SortDescriptor>)
	GetEnumerator	(Inherited from System.Collections.ObjectModel.Collection<SortDescriptor>)
	IndexOf	(Inherited from System.Collections.ObjectModel.Collection<SortDescriptor>)
	Insert	(Inherited from System.Collections.ObjectModel.Collection<SortDescriptor>)
	Move	(Inherited from System.Collections.ObjectModel.ObservableCollection<SortDescriptor>)
	Remove	(Inherited from System.Collections.ObjectModel.Collection<SortDescriptor>)
	RemoveAt	(Inherited from System.Collections.ObjectModel.Collection<SortDescriptor>)

Top

Public Events

	Name	Description
	CollectionChanged	(Inherited from System.Collections.ObjectModel.ObservableCollection<SortDescriptor>)

Top

See Also

Reference

SortDescriptorCollection Class
C1.Data.DataSource Namespace

SortDescriptorCollection Constructor
Initializes a new instance of the SortDescriptorCollection class.

Syntax

Visual Basic (Declaration)
`Public Function New()`
C#
`public SortDescriptorCollection()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SortDescriptorCollection Class
SortDescriptorCollection Members

Enumerations

FilterDescriptorLogicalOperator
Enumeration of logical operators for filter collections.

Syntax

Visual Basic (Declaration)
`Public Enum FilterDescriptorLogicalOperator` `  Inherits System.Enum`
C#
`public enum FilterDescriptorLogicalOperator : System.Enum`

Members

Member	Description
And	Filters are AND'ed.
Or	Filters are OR'ed.

Inheritance Hierarchy

System.Object
  System.ValueType
    System.Enum
      C1.Data.DataSource.FilterDescriptorLogicalOperator

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1.Data.DataSource Namespace

FilterOperator
Operator used in FilterDescriptor class.

Syntax

Visual Basic (Declaration)
`Public Enum FilterOperator` `  Inherits System.Enum`
C#
`public enum FilterOperator : System.Enum`

Members

Member	Description
Contains	Left operand must contain the right one.
EndsWith	Left operand must end with the right one.
IsContainedIn	Left operand must be contained in the right one.
IsEqualTo	Left operand must be equal to the right one.
IsGreaterThan	Left operand must be larger than or equal to the right one.
IsGreaterThanOrEqualTo	Left operand must be larger than the right one.
IsLessThan	Left operand must be smaller than the right one.
IsLessThanOrEqualTo	Left operand must be smaller than or equal to the right one.
IsNotEqualTo	Left operand must be different from the right one.
StartsWith	// Left operand must start with the right one.

Inheritance Hierarchy

System.Object
  System.ValueType
    System.Enum
      C1.Data.DataSource.FilterOperator

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1.Data.DataSource Namespace

VirtualModeKind

Enumeration of possible virtual modes a ClientViewSource can be in. Used in the ClientViewSource.VirtualMode property.

Syntax

Visual Basic (Declaration)
`Public Enum VirtualModeKind` `  Inherits System.Enum`
C#
`public enum VirtualModeKind : System.Enum`

Members

Member	Description
Managed	Virtual mode is managed by a GUI control bound to the ClientViewSource. That GUI control must have a control handler with the BaseControlHandler.VirtualMode property set to True.
None	Virtual mode is disabled.
Unmanaged	Virtual mode is not managed by a control handler, it is managed by the ClientViewSource itself that is unaware of what controls are bound to it. This option should be used only if you don't have a GUI control that supports virtual mode through a control handler. Although it allows to use virtual mode with any GUI bound control (with or without a control handler), it should be used with caution, only if you can't use the Managed option. See the "Programming Guide" in the Studio for Entity Framework documentation for more details.

Inheritance Hierarchy

System.Object
  System.ValueType

System.Enum

C1.Data.DataSource.VirtualModeKind

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1.Data.DataSource Namespace

C1.Data.Entities Namespace

Overview

Classes

	Class	Description
	EntityClientCache	Represents a client-side cache specific to Entity Framework.
	EntityClientScope	Defines a scope of data access. Provides facilities to create client views.
	EntityFrameworkExtensions	Provides a set of extensions methods for Entity Framework.
	EntityViewSource	An Entity Framework-specific version of the C1.Data.DataSource.ClientViewSource class.

See Also

Reference

C1.Data.Entity.4 Assembly

Classes

EntityClientCache

Represents a client-side cache specific to Entity Framework.

Object Model

EntityClientCache

Syntax

Visual Basic (Declaration)
`Public Class EntityClientCache` `  Inherits C1.Data.ClientCacheBase`
C#
`public class EntityClientCache : C1.Data.ClientCacheBase`

Remarks

Usually, a single instance of this class is created on application startup and exists during the entire application lifetime, while each form, window, or user control works with data using a EntityClientScope created by calling the CreateScope method.

Inheritance Hierarchy

System.Object
  C1.Data.ClientCacheBase
    C1.Data.Entities.EntityClientCache

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

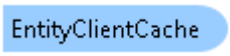
See Also

Reference

EntityClientCache Members
C1.Data.Entities Namespace

Overview
Represents a client-side cache specific to Entity Framework.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class EntityClientCache` `  Inherits C1.Data.ClientCacheBase`
C#
`public class EntityClientCache : C1.Data.ClientCacheBase`

Remarks

Usually, a single instance of this class is created on application startup and exists during the entire application lifetime, while each form, window, or user control works with data using a EntityClientScope created by calling the CreateScope method.

Inheritance Hierarchy

System.Object
  C1.Data.ClientCacheBase
    C1.Data.Entities.EntityClientCache

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

EntityClientCache Members
C1.Data.Entities Namespace

Members
Properties Methods

The following tables list the members exposed by EntityClientCache.

Public Constructors

	Name	Description
	EntityClientCache Constructor	Initializes a new instance of the EntityClientCache class.

Top

Public Properties

	Name	Description
	ObjectContext	The ObjectContext through which EntityClientCache accesses the data.

Top

Public Methods

	Name	Description
	AcceptChanges	Accepts all changes made to entities in the ObjectContext.
	BulkChanges	Used to group massive changes to entities and to allow manual explicit changes of entity states. (Inherited from C1.Data.ClientCacheBase)
	CleanupCache	Forces unused memory to be released, unused entities to be detached from the context. It is usually done automatically, so programmers rarely need to call this method in code. (Inherited from C1.Data.ClientCacheBase)
	Clear	Clears client-side cache entirely. Call this method if you want to make sure that following queries will fetch fresh data from the server. (Inherited from C1.Data.ClientCacheBase)
	CreateScope	Creates an Entity Framework-specific client scope.
	CreateTransact	Creates a C1.Data.Transactions.ClientTransaction that allows you to easily

	ion	cancel changes made in transaction scope. (Inherited from C1.Data.ClientCacheBase)
S	GetDefault	Returns the default EntityClientCache for a given contextType.
	Refresh	Refreshes data in all C1DataSource controls connected to this ClientCacheBase. (Inherited from C1.Data.ClientCacheBase)
S	RegisterContext	Overloaded. Registers an ObjectContext as a default for C1DataSource controls for a given **context type**.
	RejectChanges	Reverts all pending changes for this C1.Data.ClientCacheBase. It is recommended to call this method instead of System.Data.Objects.ObjectContext.Refresh(System.Data.Objects.RefreshMode,System.Object). (Inherited from C1.Data.ClientCacheBase)
	SaveChanges	Persists all changes to the server. It is recommended to call this method instead of System.Data.Objects.ObjectContext.SaveChanges. (Inherited from C1.Data.ClientCacheBase)

Top

See Also

Reference

EntityClientCache Class
C1.Data.Entities Namespace

EntityClientCache Constructor
The object context that is used to access the data.

Initializes a new instance of the EntityClientCache class.

Syntax

Visual Basic (Declaration)
`Public Function New( _`

```
   ByVal baseContext As ObjectContext _
)
```

C#

```
public EntityClientCache(
   ObjectContext baseContext
)
```

Parameters

baseContext

The object context that is used to access the data.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

EntityClientCache Class
EntityClientCache Members

Methods
For a list of all members of this type, see EntityClientCache members.

Public Methods

	Name	Description
	AcceptChanges	Accepts all changes made to entities in the ObjectContext.
	BulkChanges	Used to group massive changes to entities and to allow manual explicit changes of entity states. (Inherited from C1.Data.ClientCacheBase)
	CleanupCache	Forces unused memory to be released, unused entities to be detached from

		the context. It is usually done automatically, so programmers rarely need to call this method in code. (Inherited from C1.Data.ClientCacheBase)
	Clear	Clears client-side cache entirely. Call this method if you want to make sure that following queries will fetch fresh data from the server. (Inherited from C1.Data.ClientCacheBase)
	CreateScope	Creates an Entity Framework-specific client scope.
	CreateTransaction	Creates a C1.Data.Transactions.ClientTransaction that allows you to easily cancel changes made in transaction scope. (Inherited from C1.Data.ClientCacheBase)
S	GetDefault	Returns the default EntityClientCache for a given contextType.
	Refresh	Refreshes data in all C1DataSource controls connected to this ClientCacheBase. (Inherited from C1.Data.ClientCacheBase)
S	RegisterContext	Overloaded. Registers an ObjectContext as a default for C1DataSource controls for a given **context type**.
	RejectChanges	Reverts all pending changes for this C1.Data.ClientCacheBase. It is recommended to call this method instead of System.Data.Objects.ObjectContext.Refresh(System.Data.Objects.RefreshMode,System.Object). (Inherited from C1.Data.ClientCacheBase)
	SaveChanges	Persists all changes to the server. It is recommended to call this method instead of System.Data.Objects.ObjectContext.SaveChanges. (Inherited from C1.Data.ClientCacheBase)

Top

See Also

Reference

EntityClientCache Class
C1.Data.Entities Namespace

AcceptChanges Method
Accepts all changes made to entities in the ObjectContext.

Syntax

Visual Basic (Declaration)
`Public Sub AcceptChanges()`
C#
`public void AcceptChanges()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

EntityClientCache Class
EntityClientCache Members

CreateScope Method
Creates an Entity Framework-specific client scope.

Syntax

Visual Basic (Declaration)
`Public Shadows Function CreateScope() As EntityClientScope`
C#
`public new EntityClientScope CreateScope()`

Return Value

A new client scope.

Remarks

Usually, each window, form, or user control creates a C1.Data.ClientScope and uses it to access entities.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

EntityClientCache Class
EntityClientCache Members

GetDefault Method
A subclass of ObjectContext to get the default EntityClientCache for.

Returns the default EntityClientCache for a given contextType.

Syntax

Visual Basic (Declaration)

`Public Shared Function GetDefault( _`
`  ByVal contextType As Type _`
`) As EntityClientCache`

C#

`public static EntityClientCache GetDefault(`
`  Type contextType`
`)`

Parameters

contextType

A subclass of ObjectContext to get the default EntityClientCache for.

Return Value

The default EntityClientCache for the given contextType.

Remarks

Creates an EntityClientCache for the specified contextType if it does not already exist; otherwise, returns an existing instance.

It is the same default client cache as used by C1DataSource with specified C1DataSource.ObjectContextType.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

EntityClientCache Class
EntityClientCache Members

RegisterContext Method
Registers an ObjectContext as a default for C1DataSource controls for a given **context type**.

Overload List

Overload	Description
RegisterContext(ObjectContext,Type)	Registers an ObjectContext as a default for C1DataSource controls for a given contextType.
RegisterContext(ObjectContext)	Registers an ObjectContext as a default for C1DataSource controls.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

EntityClientCache Class
EntityClientCache Members

RegisterContext(ObjectContext,Type) Method
An ObjectContext to set as default.

The type (derived from ObjectContext) to register the context for.

Registers an ObjectContext as a default for C1DataSource controls for a given contextType.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function RegisterContext( _
   ByVal context As ObjectContext, _
   ByVal contextType As Type _
) As IDisposable
```

C#

```
public static IDisposable RegisterContext(
   ObjectContext context,
   Type contextType
)
```

Parameters

context

An ObjectContext to set as default.

contextType

The type (derived from ObjectContext) to register the context for.

Return Value

An System.IDisposable to unregister the context.

Exceptions

Exception	Description
System.InvalidOperationException	Another context is already registered for the given contextType.

Remarks

Use this method when you need to customize the default ObjectContext used in C1DataSource controls. Register a custom ObjectContext on startup before any C1DataSource instances are created.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

EntityClientCache Class
EntityClientCache Members
Overload List
GetDefault Method

RegisterContext(ObjectContext) Method
An ObjectContext to set as default.

Registers an ObjectContext as a default for C1DataSource controls.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function RegisterContext( _
   ByVal context As ObjectContext _
) As IDisposable
```

C#

```
public static IDisposable RegisterContext(
   ObjectContext context
```

```
)
```

Parameters

context

An ObjectContext to set as default.

Return Value

An System.IDisposable to unregister the context.

Exceptions

Exception	Description
System.InvalidOperationException	Another context is already registered.

Remarks

Use this method when you need to customize the default ObjectContext used in C1DataSource controls. Register a custom ObjectContext on startup before any C1DataSource instances are created.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

EntityClientCache Class
EntityClientCache Members
Overload List
GetDefault Method

Properties
For a list of all members of this type, see EntityClientCache members.

Public Properties

	Name	Description
	ObjectContext	The ObjectContext through which EntityClientCache accesses the data.

Top

See Also

Reference

EntityClientCache Class
C1.Data.Entities Namespace

ObjectContext Property
The ObjectContext through which EntityClientCache accesses the data.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property ObjectContext As ObjectContext`
C#
`public ObjectContext ObjectContext {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

EntityClientCache Class
EntityClientCache Members

EntityClientScope
Defines a scope of data access. Provides facilities to create client views.

Object Model

EntityClientScope

Syntax

Visual Basic (Declaration)
`Public Class EntityClientScope` `  Inherits C1.Data.ClientScope`
C#
`public class EntityClientScope : C1.Data.ClientScope`

Remarks

Usually, one scope is created for each window/user control, and disposed at the end of its lifetime. Entities pinned to the scope (marked as needed) are not evicted from the cache until the scope is disposed or collected by the GC.

Inheritance Hierarchy

System.Object
  C1.Data.ClientScope
    C1.Data.Entities.EntityClientScope

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

EntityClientScope Members
C1.Data.Entities Namespace

Overview
Defines a scope of data access. Provides facilities to create client views.

Object Model

EntityClientScope

Syntax

Visual Basic (Declaration)
`Public Class EntityClientScope` `  Inherits C1.Data.ClientScope`
C#
`public class EntityClientScope : C1.Data.ClientScope`

Remarks

Usually, one scope is created for each window/user control, and disposed at the end of its lifetime. Entities pinned to the scope (marked as needed) are not evicted from the cache until the scope is disposed or collected by the GC.

Inheritance Hierarchy

System.Object
  C1.Data.ClientScope
    C1.Data.Entities.EntityClientScope

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

EntityClientScope Members
C1.Data.Entities Namespace

Members
Properties Methods

The following tables list the members exposed by EntityClientScope.

Public Constructors

	Name	Description
	EntityClientScope Constructor	Initializes a new instance of the EntityClientScope class with the specified EntityClientCache.

Top

Public Properties

	Name	Description
	ClientCache	Gets the EntityClientCache to which this client scope is connected.

Top

Public Methods

	Name	Description
	AddRef	Overloaded. Marks an entity as needed. Needed entities are not detached/released from the context until the client scope is disposed. (Inherited from C1.Data.ClientScope)
	Dispose	Marks the scope as disposed. Entities that were marked needed by a disposed scope may be disposed of (evicted from the cache, detached from context) unless they are needed by other active scopes. (Inherited from C1.Data.ClientScope)
	GetItems	Overloaded. Gets a client view of entities of a given type.
	Release	Overloaded. Unmark a needed entity. (Inherited from C1.Data.ClientScope)

Top

See Also

Reference

EntityClientScope Class
C1.Data.Entities Namespace

EntityClientScope Constructor
An instance of the EntityClientCache class to which the new client scope is connected.

Initializes a new instance of the EntityClientScope class with the specified EntityClientCache.

Syntax

Visual Basic (Declaration)

```
Public Function New( _
   ByVal clientCache As EntityClientCache _
)
```

C#

```
public EntityClientScope(
   EntityClientCache clientCache
)
```

Parameters

clientCache

An instance of the EntityClientCache class to which the new client scope is connected.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

EntityClientScope Class
EntityClientScope Members

Methods
For a list of all members of this type, see EntityClientScope members.

Public Methods

	Name	Description
	AddRef	Overloaded. Marks an entity as needed. Needed entities are not detached/released from the context until the client scope is disposed. (Inherited from C1.Data.ClientScope)
	Dispose	Marks the scope as disposed. Entities that were marked needed by a disposed scope may be disposed of (evicted from the cache, detached from context) unless they are needed by other active scopes. (Inherited from C1.Data.ClientScope)
	GetItems	Overloaded. Gets a client view of entities of a given type.
	Release	Overloaded. Unmark a needed entity. (Inherited from C1.Data.ClientScope)

Top

See Also

Reference

EntityClientScope Class
C1.Data.Entities Namespace

GetItems Method
Gets a client view of entities of a given type.

Overload List

Overload	Description
GetItems<T>()	Gets a client view of entities of a given type.
GetItems<T>(String)	Gets a client view of entities from the specified entitySetName.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

EntityClientScope Class
EntityClientScope Members

GetItems<T>() Method
The type of entities to load.

Gets a client view of entities of a given type.

Syntax

Visual Basic (Declaration)
`Public Overloads Function GetItems(Of T)() As ClientView(Of T)`
C#
`public ClientView<T> GetItems<T>()`

Type Parameters

T

The type of entities to load.

Return Value

A client view of entities of the specified type.

Remarks

Entities are loaded using the entity set of the matching entity type from the default entity container.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

EntityClientScope Class
EntityClientScope Members
Overload List

GetItems<T>(String) Method
The type of entities in the entitySetName.

The name of the entity set to load entities from.

Gets a client view of entities from the specified entitySetName.

Syntax

Visual Basic (Declaration)

`Public Overloads Function GetItems(Of T)( _`
`  ByVal entitySetName As String _`
`) As ClientView(Of T)`

C#

`public ClientView<T> GetItems<T>(`
`  string entitySetName`
`)`

Parameters

entitySetName

The name of the entity set to load entities from.

Type Parameters

T

The type of entities in the entitySetName.

Return Value

A client view of entities from the specified entitySetName.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

EntityClientScope Class
EntityClientScope Members
Overload List

Properties
For a list of all members of this type, see EntityClientScope members.

Public Properties

	Name	Description
	ClientCache	Gets the EntityClientCache to which this client scope is connected.

Top

See Also

Reference

EntityClientScope Class
C1.Data.Entities Namespace

ClientCache Property
Gets the EntityClientCache to which this client scope is connected.

Syntax

Visual Basic (Declaration)

```
Public Shadows ReadOnly Property ClientCache As EntityClientCache
```

C#

```
public new EntityClientCache ClientCache {get;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

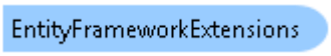
See Also

Reference

EntityClientScope Class
EntityClientScope Members

EntityFrameworkExtensions
Provides a set of extensions methods for Entity Framework.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit NotInheritable Class EntityFrameworkExtensions`
C#
`public static class EntityFrameworkExtensions`

Inheritance Hierarchy

System.Object
  C1.Data.Entities.EntityFrameworkExtensions

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

EntityFrameworkExtensions Members
C1.Data.Entities Namespace

Overview
Provides a set of extensions methods for Entity Framework.

Object Model

EntityFrameworkExtensions

Syntax

Visual Basic (Declaration)
`Public MustInherit NotInheritable Class EntityFrameworkExtensions`
C#
`public static class EntityFrameworkExtensions`

Inheritance Hierarchy

System.Object
  C1.Data.Entities.EntityFrameworkExtensions

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

EntityFrameworkExtensions Members
C1.Data.Entities Namespace

Members
Methods

The following tables list the members exposed by EntityFrameworkExtensions.

Public Methods

	Name	Description
S	AsCollectionView<T>	Converts an EntityCollection to an editable

		System.ComponentModel.ICollectionView.
S	AsLive	Overloaded. Converts an EntityCollection to an editable live view.

Top

See Also

Reference

EntityFrameworkExtensions Class
C1.Data.Entities Namespace

Methods
>

Name	Description
S AsCollectionView<T>	Converts an EntityCollection to an editable System.ComponentModel.ICollectionView.
S AsLive	Overloaded. Converts an EntityCollection to an editable live view.

Top

See Also

Reference

EntityFrameworkExtensions Class
C1.Data.Entities Namespace

AsCollectionView<T> Method
The type of the entities in the entities.

The EntityCollection to convert.

Converts an EntityCollection to an editable System.ComponentModel.ICollectionView.

Syntax

Visual Basic (Declaration)
`Public Shared Function AsCollectionView(Of T As {Class,`

```
IEntityWithRelationships})( _
   ByVal entities As EntityCollection(Of T) _
) As ICollectionView
```

C#

```
public static ICollectionView AsCollectionView<T>(
   EntityCollection<T> entities
)
where T: class, IEntityWithRelationships
```

Parameters

entities

The EntityCollection to convert.

Type Parameters

T

The type of the entities in the entities.

Return Value

The resulting System.ComponentModel.ICollectionView.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

EntityFrameworkExtensions Class
EntityFrameworkExtensions Members

AsLive Method
Converts an EntityCollection to an editable live view.

Overload List

Overload	Description
AsLive<T>(EntityCollection<T>)	Converts an EntityCollection to an editable live view.
AsLive<T>(ICollection<T>,EntityClientScope)	Converts a POCO EntityCollection to an editable live view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

EntityFrameworkExtensions Class
EntityFrameworkExtensions Members

AsLive<T>(EntityCollection<T>) Method
The type of the entities in the entities.

The entity collection to convert.

Converts an EntityCollection to an editable live view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function AsLive(Of T As {Class,
IEntityWithRelationships})( _
   ByVal entities As EntityCollection(Of T) _
) As View(Of T)
```

C#

```
public static View<T> AsLive<T>(
   EntityCollection<T> entities
)
where T: class, IEntityWithRelationships
```

Parameters

entities

The entity collection to convert.

Type Parameters

T

The type of the entities in the entities.

Return Value

The resulting live view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

EntityFrameworkExtensions Class
EntityFrameworkExtensions Members
Overload List

AsLive<T>(ICollection<T>,EntityClientScope) Method
The type of the entities in the pocoCollection.

The entity collection to convert. It must be of type EntityCollection.

The EntityClientScope to which the entity owning this collection belongs (in which it was fetched by a query or created).

Converts a POCO EntityCollection to an editable live view.

Syntax

Visual Basic (Declaration)	
`Public Overloads Shared Function AsLive(Of T As Class)( _` `  ByVal pocoCollection As ICollection(Of T), _`	

```
   ByVal scope As EntityClientScope _
) As View(Of T)
```

C#

```
public static View<T> AsLive<T>(
   ICollection<T> pocoCollection,
   EntityClientScope scope
)
where T: class
```

Parameters

pocoCollection

The entity collection to convert. It must be of type EntityCollection.

scope

The EntityClientScope to which the entity owning this collection belongs (in which it was fetched by a query or created).

Type Parameters

T

The type of the entities in the pocoCollection.

Return Value

The resulting live view.

Exceptions

Exception	Description
System.ArgumentException	The pocoCollection is not of type EntityCollection.

Remarks

When POCO objects are used (with proxies), navigation collection properties are typed as ICollection, not EntityCollection (although they are EntityCollection at run time). That is why a special AsLive extension method must be used.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

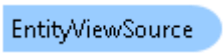
See Also

Reference

EntityFrameworkExtensions Class
EntityFrameworkExtensions Members
Overload List

EntityViewSource
An Entity Framework-specific version of the C1.Data.DataSource.ClientViewSource class.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class EntityViewSource` `  Inherits C1.Data.DataSource.ClientViewSource`
C#
`public class EntityViewSource : C1.Data.DataSource.ClientViewSource`

Remarks

To load data, specify the name of an entity set to load the data from.

Inheritance Hierarchy

System.Object
  System.Windows.Threading.DispatcherObject
    System.Windows.DependencyObject
      C1.Data.DataSource.ClientViewSource
        C1.Data.Entities.EntityViewSource

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

EntityViewSource Members
C1.Data.Entities Namespace

Overview
An Entity Framework-specific version of the C1.Data.DataSource.ClientViewSource class.

Object Model

Syntax

Visual Basic (Declaration)
`Public Class EntityViewSource` `  Inherits C1.Data.DataSource.ClientViewSource`
C#
`public class EntityViewSource : C1.Data.DataSource.ClientViewSource`

Remarks

To load data, specify the name of an entity set to load the data from.

Inheritance Hierarchy

System.Object
  System.Windows.Threading.DispatcherObject
    System.Windows.DependencyObject
      C1.Data.DataSource.ClientViewSource
        C1.Data.Entities.EntityViewSource

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

EntityViewSource Members
C1.Data.Entities Namespace

Members
Properties Methods Events

The following tables list the members exposed by EntityViewSource.

Public Constructors

	Name	Description
	EntityViewSource Constructor	

Top

Public Properties

	Name	Description
	AutoLoad	Gets or sets a value indicating whether Load is automatically invoked on startup and when a change occurs that impacts the query composed by the C1.Data.DataSource.ClientViewSource. The default is True. (Inherited from C1.Data.DataSource.ClientViewSource)
	BaseView	Gets or sets an instance of C1.Data.ClientView<T> that the C1.Data.DataSource.ClientViewSource uses as the base for composing queries. (Inherited from C1.Data.DataSource.ClientViewSource)
	CacheTimeout	Gets or sets the period of time entities loaded in virtual mode are kept in the cache without checking whether they are needed or not. If an entity was neither used nor considered needed for a period of time

		longer than CacheTimeout, C1.Data.DataSource.ClientViewSource may evict it from the cache. (Inherited from C1.Data.DataSource.ClientViewSource)
	CurrentClientView	Gets the current client view used to load entities, or null in virtual mode. (Inherited from C1.Data.DataSource.ClientViewSource)
	DataView	Gets the current view of entities resulting from the last load operation. (Inherited from C1.Data.DataSource.ClientViewSource)
	DependencyObjectType	(Inherited from System.Windows.DependencyObject)
	Dispatcher	(Inherited from System.Windows.Threading.DispatcherObject)
	EntitySetName	Gets or sets the name of the entity set to load entities from.
	FilterDescriptors	Gets the collection of C1.Data.DataSource.FilterDescriptor objects used when performing loads. (Inherited from C1.Data.DataSource.ClientViewSource)
	FilterOperator	Gets or sets the logical operator used for combinining FilterDescriptors in the filter collection. The default value is C1.Data.DataSource.FilterDescriptorLogicalOperator.And. (Inherited from C1.Data.DataSource.ClientViewSource)
	GroupDescriptors	Gets the collection of C1.Data.DataSource.GroupDescriptor objects used to organize the loaded entities into groups. (Inherited from C1.Data.DataSource.ClientViewSource)
	Include	Gets or sets a comma-separated list of property paths that specify related objects to include during the Load operation. (Inherited from C1.Data.DataSource.ClientViewSource)
	IsLoadingData	Gets a value indicating whether the C1.Data.DataSource.ClientViewSource is currently loading data.

		(Inherited from C1.Data.DataSource.ClientViewSource)
	IsSealed	(Inherited from System.Windows.DependencyObject)
	LoadCommand	Gets an System.Windows.Input.ICommand that invokes Load on this C1.Data.DataSource.ClientViewSource. (Inherited from C1.Data.DataSource.ClientViewSource)
	LoadDelay	Gets or sets the delay before an automatic data loading operation is started. It is the delay from the time a change prompting automatic load occurs until the time the resulting Load is started. The default delay is 25 milliseconds. (Inherited from C1.Data.DataSource.ClientViewSource)
	LoadSize	Gets or sets the maximum number of items to load each time a Load is executed. When equal to 0, all requested entities will be loaded. The default is 0. (Inherited from C1.Data.DataSource.ClientViewSource)
	MoveToFirstOnLoad	Gets or sets a value indicating that the first item must be made current after Load operation is completed if current item was not set by other means. (Inherited from C1.Data.DataSource.ClientViewSource)
	Name	Overridden. Gets a name of this EntityViewSource to reference it in a C1DataSource.ViewSources collection. It is determined by the EntitySetName but can be overrided by the NameOverride.
	NameOverride	Gets or sets a value that overrides the value of the Name property. (Inherited from C1.Data.DataSource.ClientViewSource)
	PageSize	Gets or sets the number of items displayed on each page of the DataView, or the number of items to fetch in each query in virtual mode, or 0 to disable paging. (Inherited from C1.Data.DataSource.ClientViewSource)

	SortDescriptors	Gets the collection of C1.Data.DataSource.SortDescriptor objects used to sort the data. (Inherited from C1.Data.DataSource.ClientViewSource)
	VirtualMode	Gets or sets a value indicating whether the C1.Data.DataSource.ClientViewSource is in virtual mode. Virtual mode is an innovative technology allowing to bind GUI controls directly to very large data sets without delays and performance degradation and without inconvenience of paging. By default, virtual mode is disabled (the default value is C1.Data.DataSource.VirtualModeKind.None). (Inherited from C1.Data.DataSource.ClientViewSource)

Top

Public Methods

	Name	Description
	ClearValue	Overloaded. (Inherited from System.Windows.DependencyObject)
	CoerceValue	(Inherited from System.Windows.DependencyObject)
	DeferLoad	Used to group changes to multiple load-affecting properties together, deferring the resulting load operations so a single load operation is performed in the end, that is, when the object returned from this method is disposed. (Inherited from C1.Data.DataSource.ClientViewSource)
	Equals	(Inherited from System.Windows.DependencyObject)
	GetHashCode	(Inherited from System.Windows.DependencyObject)
	GetLocalValueEnumerator	(Inherited from System.Windows.DependencyObject)
	GetValue	(Inherited from System.Windows.DependencyObject)

	InvalidateProperty	(Inherited from System.Windows.DependencyObject)
	Load	Starts a load operation. Any pending load will be implicitly canceled. (Inherited from C1.Data.DataSource.ClientViewSource)
	LoadRange	If in virtual mode, loads a specific range of entities. (Inherited from C1.Data.DataSource.ClientViewSource)
	ReadLocalValue	(Inherited from System.Windows.DependencyObject)
	Refresh	Starts a load operation ignoring the client-side cache. Any pending load will be implicitly canceled. (Inherited from C1.Data.DataSource.ClientViewSource)
	SetCurrentValue	(Inherited from System.Windows.DependencyObject)
	SetValue	Overloaded. (Inherited from System.Windows.DependencyObject)

Top

Public Events

	Name	Description
	LoadedData	Occurs when a load operation is completed, or when an exception was thrown during the load operation. (Inherited from C1.Data.DataSource.ClientViewSource)
	PropertyChanged	Occurs when a property value changes. (Inherited from C1.Data.DataSource.ClientViewSource)

Top

See Also

Reference

EntityViewSource Class
C1.Data.Entities Namespace

EntityViewSource Constructor

Syntax

Visual Basic (Declaration)
`Public Function New()`
C#
`public EntityViewSource()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

EntityViewSource Class
EntityViewSource Members

Properties
For a list of all members of this type, see EntityViewSource members.

Public Properties

	Name	Description
	AutoLoad	Gets or sets a value indicating whether Load is automatically invoked on startup and when a change occurs that impacts the query composed by the C1.Data.DataSource.ClientViewSource. The default is True. (Inherited from C1.Data.DataSource.ClientViewSource)
	BaseView	Gets or sets an instance of C1.Data.ClientView<T> that the C1.Data.DataSource.ClientViewSource uses as the base for composing

		queries. (Inherited from C1.Data.DataSource.ClientViewSource)
	CacheTimeout	Gets or sets the period of time entities loaded in virtual mode are kept in the cache without checking whether they are needed or not. If an entity was neither used nor considered needed for a period of time longer than CacheTimeout, C1.Data.DataSource.ClientViewSource may evict it from the cache. (Inherited from C1.Data.DataSource.ClientViewSource)
	CurrentClientView	Gets the current client view used to load entities, or null in virtual mode. (Inherited from C1.Data.DataSource.ClientViewSource)
	DataView	Gets the current view of entities resulting from the last load operation. (Inherited from C1.Data.DataSource.ClientViewSource)
	DependencyObjectType	(Inherited from System.Windows.DependencyObject)
	Dispatcher	(Inherited from System.Windows.Threading.DispatcherObject)
	EntitySetName	Gets or sets the name of the entity set to load entities from.
	FilterDescriptors	Gets the collection of C1.Data.DataSource.FilterDescriptor objects used when performing loads. (Inherited from C1.Data.DataSource.ClientViewSource)
	FilterOperator	Gets or sets the logical operator used for combinining FilterDescriptors in the filter collection. The default value is C1.Data.DataSource.FilterDescriptorLogicalOperator.And. (Inherited from C1.Data.DataSource.ClientViewSource)
	GroupDescriptors	Gets the collection of C1.Data.DataSource.GroupDescriptor objects used to organize the loaded entities into groups. (Inherited from C1.Data.DataSource.ClientViewSource)

	Include	Gets or sets a comma-separated list of property paths that specify related objects to include during the Load operation. (Inherited from C1.Data.DataSource.ClientViewSource)
	IsLoadingData	Gets a value indicating whether the C1.Data.DataSource.ClientViewSource is currently loading data. (Inherited from C1.Data.DataSource.ClientViewSource)
	IsSealed	(Inherited from System.Windows.DependencyObject)
	LoadCommand	Gets an System.Windows.Input.ICommand that invokes Load on this C1.Data.DataSource.ClientViewSource. (Inherited from C1.Data.DataSource.ClientViewSource)
	LoadDelay	Gets or sets the delay before an automatic data loading operation is started. It is the delay from the time a change prompting automatic load occurs until the time the resulting Load is started. The default delay is 25 milliseconds. (Inherited from C1.Data.DataSource.ClientViewSource)
	LoadSize	Gets or sets the maximum number of items to load each time a Load is executed. When equal to 0, all requested entities will be loaded. The default is 0. (Inherited from C1.Data.DataSource.ClientViewSource)
	MoveToFirstOnLoad	Gets or sets a value indicating that the first item must be made current after Load operation is completed if current item was not set by other means. (Inherited from C1.Data.DataSource.ClientViewSource)
	Name	Overridden. Gets a name of this EntityViewSource to reference it in a C1DataSource.ViewSources collection. It is determined by the EntitySetName but can be overrided by the NameOverride.
	NameOverride	Gets or sets a value that overrides the value of the Name property.

		(Inherited from C1.Data.DataSource.ClientViewSource)
	PageSize	Gets or sets the number of items displayed on each page of the DataView, or the number of items to fetch in each query in virtual mode, or 0 to disable paging. (Inherited from C1.Data.DataSource.ClientViewSource)
	SortDescriptors	Gets the collection of C1.Data.DataSource.SortDescriptor objects used to sort the data. (Inherited from C1.Data.DataSource.ClientViewSource)
	VirtualMode	Gets or sets a value indicating whether the C1.Data.DataSource.ClientViewSource is in virtual mode. Virtual mode is an innovative technology allowing to bind GUI controls directly to very large data sets without delays and performance degradation and without inconvenience of paging. By default, virtual mode is disabled (the default value is C1.Data.DataSource.VirtualModeKind.None). (Inherited from C1.Data.DataSource.ClientViewSource)

Top

See Also

Reference

EntityViewSource Class
C1.Data.Entities Namespace

EntitySetName Property
Gets or sets the name of the entity set to load entities from.

Syntax

Visual Basic (Declaration)
`Public Property EntitySetName As String`
C#

```
public string EntitySetName {get; set;}
```

Remarks

Changing the value of this property causes the EntityViewSource to reload data if AutoLoad is set to true.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

EntityViewSource Class
EntityViewSource Members

Name Property
Gets a name of this EntityViewSource to reference it in a C1DataSource.ViewSources collection. It is determined by the EntitySetName but can be overrided by the NameOverride.

Syntax

Visual Basic (Declaration)
`Public Overrides ReadOnly Property Name As String`
C#
`public override string Name {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

EntityViewSource Class
EntityViewSource Members

C1.Data.Transactions Namespace

Overview

Classes

	Class	Description
	ClientTransaction	Represents a transaction that tracks client-side changes and can roll them back.

See Also

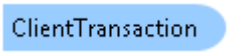
Reference

C1.Data.Entity.4 Assembly

Classes

ClientTransaction
Represents a transaction that tracks client-side changes and can roll them back.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class ClientTransaction` `  Implements C1.LiveLinq.ITransaction`
C#
`public class ClientTransaction : C1.LiveLinq.ITransaction`

Remarks

To create a new independent transaction, use the C1.Data.ClientCacheBase.CreateTransaction method. To create a child transaction, use the constructor.

Inheritance Hierarchy

System.Object
  C1.Data.Transactions.ClientTransaction

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

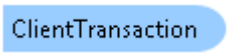
See Also

Reference

ClientTransaction Members
C1.Data.Transactions Namespace

Overview
Represents a transaction that tracks client-side changes and can roll them back.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class ClientTransaction` `  Implements C1.LiveLinq.ITransaction`
C#
`public class ClientTransaction : C1.LiveLinq.ITransaction`

Remarks

To create a new independent transaction, use the C1.Data.ClientCacheBase.CreateTransaction method. To create a child transaction, use the constructor.

Inheritance Hierarchy

System.Object
C1.Data.Transactions.ClientTransaction

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientTransaction Members
C1.Data.Transactions Namespace

Members
Properties Methods Events

The following tables list the members exposed by ClientTransaction.

Public Constructors

	Name	Description
	ClientTransaction Constructor	Initializes a child (nested) transaction, a new instance of the ClientTransaction class with a specified parent transaction.

Top

Public Properties

	Name	Description
	HasChanges	Gets a value indicating whether any changes were made in the scope of this transaction.
	State	Gets the state the transaction is in.

Top

Public Methods

	Name	Description
	Commit	Commits the transaction if it was not committed before. Commits changes that were made while this transaction's scope was open.
	Dispose	Disposes of the ClientTransaction.
	Rollback	Rolls back the transaction.
	Scope	Opens the transaction scope.
	ScopeDataContext	Wraps an object so the transaction scope is automatically opened when a value is being assigned to a property of the wrapped object.

Top

Public Events

	Name	Description
	PropertyChanged	Occurs when a property value changes, after it has been changed.

Top

See Also

Reference

ClientTransaction Class
C1.Data.Transactions Namespace

ClientTransaction Constructor
The parent transaction. Cannot be null.

Initializes a child (nested) transaction, a new instance of the ClientTransaction class with a specified parent transaction.

Syntax

Visual Basic (Declaration)	

```
Public Function New( _
   ByVal parent As ClientTransaction _
)
```

C#

```
public ClientTransaction(
   ClientTransaction parent
)
```

Parameters

parent

The parent transaction. Cannot be null.

Exceptions

Exception	Description
System.ArgumentNullException	The parent is null.

Remarks

A child transaction is automatically committed/rolled back if its parent transaction is committed/rolled back.

Create a child transaction in cases where you need to open a new window for editing a portion of data that is being editing in an already open transaction.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientTransaction Class
ClientTransaction Members

Methods

For a list of all members of this type, see ClientTransaction members.

Public Methods

	Name	Description
	Commit	Commits the transaction if it was not committed before. Commits changes that were made while this transaction's scope was open.
	Dispose	Disposes of the ClientTransaction.
	Rollback	Rolls back the transaction.
	Scope	Opens the transaction scope.
	ScopeDataContext	Wraps an object so the transaction scope is automatically opened when a value is being assigned to a property of the wrapped object.

Top

See Also

Reference

ClientTransaction Class
C1.Data.Transactions Namespace

Commit Method

Commits the transaction if it was not committed before. Commits changes that were made while this transaction's scope was open.

Syntax

Visual Basic (Declaration)
`Public Sub Commit()`
C#
`public void Commit()`

Exceptions

Exception	Description
System.InvalidOperationException	The State is not C1.LiveLinq.TransactionState.Open.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientTransaction Class
ClientTransaction Members

Dispose Method
Disposes of the ClientTransaction.

Syntax

Visual Basic (Declaration)
`Public Sub Dispose()`
C#
`public void Dispose()`

Remarks

If the State is C1.LiveLinq.TransactionState.Open, the ClientTransaction is automatically rolled back.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientTransaction Class
ClientTransaction Members

Rollback Method
Rolls back the transaction.

Syntax

Visual Basic (Declaration)	
`Public Sub Rollback()`	
C#	
`public void Rollback()`	

Exceptions

Exception	Description
System.InvalidOperationException	The State is C1.LiveLinq.TransactionState.Committed or C1.LiveLinq.TransactionState.Committing.

Remarks

Calling this method cancels the changes that were made in the scope of this transaction.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientTransaction Class
ClientTransaction Members

Scope Method
Opens the transaction scope.

Syntax

Visual Basic (Declaration)
`Public Function Scope() As IDisposable`
C#
`public IDisposable Scope()`

Return Value

An instance of an System.IDisposable that will close the scope when its System.IDisposable.Dispose method is called.

Exceptions

Exception	Description
System.InvalidOperationException	The **ClientTransaction.State** is not C1.LiveLinq.TransactionState.Open.

Remarks

The transaction tracks changes only when they are made inside an open scope.

Calling System.IDisposable.Dispose on the return value closes the scope.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientTransaction Class
ClientTransaction Members

ScopeDataContext Method
The object to wrap.

Wraps an object so the transaction scope is automatically opened when a value is being assigned to a property of the wrapped object.

Syntax

Visual Basic (Declaration)

```
Public Function ScopeDataContext( _
   ByVal entity As Object _
) As Object
```

C#

```
public object ScopeDataContext(
   object entity
)
```

Parameters

entity

The object to wrap.

Return Value

The wrapped object.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientTransaction Class
ClientTransaction Members

Properties
For a list of all members of this type, see ClientTransaction members.

Public Properties

	Name	Description
	HasChanges	Gets a value indicating whether any changes were made in the scope of this transaction.
	State	Gets the state the transaction is in.

Top

See Also

Reference

ClientTransaction Class
C1.Data.Transactions Namespace

HasChanges Property
Gets a value indicating whether any changes were made in the scope of this transaction.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Property HasChanges As Boolean
```

C#

```
public bool HasChanges {get;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientTransaction Class
ClientTransaction Members

State Property
Gets the state the transaction is in.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property State As TransactionState`
C#
`public TransactionState State {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientTransaction Class
ClientTransaction Members
TransactionState Enumeration

Events
For a list of all members of this type, see ClientTransaction members.

Public Events

	Name	Description
	PropertyChanged	Occurs when a property value changes, after it has been changed.

Top

See Also

Reference

ClientTransaction Class
C1.Data.Transactions Namespace

PropertyChanged Event
Occurs when a property value changes, after it has been changed.

Syntax

Visual Basic (Declaration)	
`Public Event PropertyChanged As PropertyChangedEventHandler`	
C#	
`public event PropertyChangedEventHandler PropertyChanged`	

Event Data

The event handler receives an argument of type PropertyChangedEventArgs containing data related to this event. The following **PropertyChangedEventArgs** properties provide information specific to this event.

Property	Description
PropertyName	

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientTransaction Class
ClientTransaction Members

C1.LiveLinq.4 Assembly

Overview

%%description%%

" -->

Namespaces

Namespace	Description
C1.LiveLinq	
C1.LiveLinq.AdoNet	
C1.LiveLinq.Collections	
C1.LiveLinq.Indexing	
C1.LiveLinq.Indexing.Search	
C1.LiveLinq.Listeners	
C1.LiveLinq.LiveViews	
C1.LiveLinq.LiveViews.Xml	
C1.LiveLinq.Metadata	
C1.WPF.LiveLinq	

Namespaces

C1.LiveLinq Namespace

Overview

Classes

	Class	Description
	CompiledQuery	Provides for compilation and cache of queries for reuse.
	DeletedStateIsAvailableAttribute	Indicates that the properties of an object of a class still return correct property values after the object has been deleted

		from the collection it belonged to.
	Hints	Provides a static (extension) method used as a hint that can be applied to an expression (usually, a property) in a query.
	IndexedQueryExtensions	Provides a set of static (extension) methods for querying objects that implement C1.LiveLinq.Indexing.IIndexedSource<T>.
	LiveViewExtensions	Provides a set of static (extension) methods used in queries to define live views.
	Ordering<T>	Represents a sorted C1.LiveLinq.Indexing.IIndexedSource<T>.
	QueryOptimizationException	Represents an exception that is thrown when Hints in a query require using a certain mandatory optimization which is impossible in the current query context.
	SourceChangeEventArgs<T>	Provides data for the IObservableSource<T>.Changed event.

Interfaces

	Interface	Description
	IObservableSource<T>	Provides methods and events that are required for LiveLinq functionality, indexing and live views.
	ITransaction	Represents a transaction with an explicit scope.

Enumerations

	Enumeration	Description
	IndexingHintAction	Specifies the actions taken by LiveLinq query optimizer when it encounters an **Indexed()** hint applied to an expression (usually, a property) in a query.

	Order	Indicates if a certain order is required in the result collection of an operation.
	SourceChangeType	Describes a change occurring in a collection.
	TransactionState	Enumeration of the possible states an ITransaction can be in.

See Also

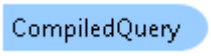
Reference

C1.LiveLinq.4 Assembly

Classes

CompiledQuery
Provides for compilation and cache of queries for reuse.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit NotInheritable Class CompiledQuery`
C#
`public static class CompiledQuery`

Remarks

If you need to execute the same query many times, each time with different parameter values, use the **CompiledQuery** class to improve performance. Without it, every query execution includes a compilation stage, that does not take much time but that time can accumulate to significant numbers if it is repeated many times. The **CompiledQuery** class contains a single **Compile** method with several overloads. Call the **Compile** method to create a delegate to represent the compiled query.

Inheritance Hierarchy

System.Object
C1.LiveLinq.CompiledQuery

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

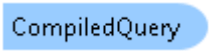
See Also

Reference

CompiledQuery Members
C1.LiveLinq Namespace

Overview
Provides for compilation and cache of queries for reuse.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit NotInheritable Class CompiledQuery`
C#
`public static class CompiledQuery`

Remarks

If you need to execute the same query many times, each time with different parameter values, use the **CompiledQuery** class to improve performance. Without it, every query execution includes a compilation stage, that does not take much time but that time can accumulate to significant numbers if it is repeated many times. The **CompiledQuery** class contains a single **Compile** method with several overloads. Call the **Compile** method to create a delegate to represent the compiled query.

Inheritance Hierarchy

System.Object
C1.LiveLinq.CompiledQuery

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

CompiledQuery Members
C1.LiveLinq Namespace

Members
Methods

The following tables list the members exposed by CompiledQuery.

Public Methods

	Name	Description
S	Compile	Overloaded. Compiles the query.

Top

See Also

Reference

CompiledQuery Class
C1.LiveLinq Namespace

Methods
>

Name	Description
S Compile	Overloaded. Compiles the query.

Top

See Also

Reference

CompiledQuery Class
C1.LiveLinq Namespace

Compile Method
Compiles the query.

Overload List

Overload	Description
Compile<T1,T2,T3,T4,TResult>(Expression<Func<T1,T2,T3,T4,IIndexedSource<TResult>>>)	Compiles the query.
Compile<T1,T2,T3,TResult>(Expression<Func<T1,T2,T3,IIndexedSource<TResult>>>)	Compiles the query.
Compile<T1,T2,TResult>(Expression<Func<T1,T2,IIndexedSource<TResult>>>)	Compiles the query.
Compile<T,TResult>(Expression<Func<T,IIndexedSource<TResult>>>)	Compiles the query.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

CompiledQuery Class
CompiledQuery Members

Compile<T1,T2,T3,T4,TResult>(Expression<Func<T1,T2,T3,T4,IIndexedSource<TResult>>>) Method
The type of the first parameter that has to be passed in when executing the delegate returned by the Compile method.

The type of the second parameter that has to be passed in when executing the delegate returned by the Compile method.

The type of the third parameter that has to be passed in when executing the delegate returned by the Compile method.

The type of the fourth parameter that has to be passed in when executing the delegate returned by the Compile method.

The type of **TResult** in the IIndexedSource<TResult> returned when executing the delegate returned by the Compile method.

The query expression to be compiled.

Compiles the query.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function Compile
    (Of T1,T2,T3,T4,TResult)( _
   ByVal query As Expression(Of Func(Of T1,T2,T3,T4,IIndexedSource(Of TResult)))
_
) As Func(Of T1,T2,T3,T4,IIndexedSource(Of TResult))
```

C#

```
public static Func<T1,T2,T3,T4,IIndexedSource<TResult>>
Compile<T1,T2,T3,T4,TResult>(
   Expression<Func<T1,T2,T3,T4,IIndexedSource<TResult>>> query
)
```

Parameters

query

The query expression to be compiled.

Type Parameters

T1

The type of the first parameter that has to be passed in when executing the delegate returned by the Compile method.

T2

The type of the second parameter that has to be passed in when executing the delegate returned by the Compile method.

T3

The type of the third parameter that has to be passed in when executing the delegate returned by the Compile method.

T4

The type of the fourth parameter that has to be passed in when executing the delegate returned by the Compile method.

TResult

The type of **TResult** in the IIndexedSource<TResult> returned when executing the delegate returned by the Compile method.

Return Value

The delegate to be called to execute the compiled query with particular parameter values.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

CompiledQuery Class
CompiledQuery Members
Overload List

The type of the first parameter that has to be passed in when executing the delegate returned by the Compile method.

The type of the second parameter that has to be passed in when executing the delegate returned by the Compile method.

The type of the third parameter that has to be passed in when executing the delegate returned by the Compile method.

The type of **TResult** in the IIndexedSource<TResult> returned when executing the delegate returned by the Compile method.

The query expression to be compiled.

Compiles the query.

Syntax

Visual Basic (Declaration)	

```
Public Overloads Shared Function Compile
    (Of T1,T2,T3,TResult)( _
   ByVal query As Expression(Of Func(Of T1,T2,T3,IIndexedSource(Of TResult))) _
) As Func(Of T1,T2,T3,IIndexedSource(Of TResult))
```

C#	

```
public static Func<T1,T2,T3,IIndexedSource<TResult>> Compile<T1,T2,T3,TResult>(
   Expression<Func<T1,T2,T3,IIndexedSource<TResult>>> query
)
```

Parameters

query

The query expression to be compiled.

Type Parameters

T1

The type of the first parameter that has to be passed in when executing the delegate returned by the Compile method.

T2

The type of the second parameter that has to be passed in when executing the delegate returned by the Compile method.

T3

The type of the third parameter that has to be passed in when executing the delegate returned by the Compile method.

TResult

The type of **TResult** in the IIndexedSource<TResult> returned when executing the delegate returned by the Compile method.

Return Value

The delegate to be called to execute the compiled query with particular parameter values.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

CompiledQuery Class
CompiledQuery Members
Overload List

Compile<T1,T2,TResult>(Expression<Func<T1,T2,IIndexedSource<TResult>>>) Method
The type of the first parameter that has to be passed in when executing the delegate returned by the Compile method.

The type of the second parameter that has to be passed in when executing the delegate returned by the Compile method.

The type of **TResult** in the IIndexedSource<TResult> returned when executing the delegate returned by the Compile method.

The query expression to be compiled.

Compiles the query.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function Compile`
`  (Of T1,T2,TResult)( _`
`  ByVal query As Expression(Of Func(Of T1,T2,IIndexedSource(Of TResult))) _`
`) As Func(Of T1,T2,IIndexedSource(Of TResult))`

C#

`public static Func<T1,T2,IIndexedSource<TResult>> Compile<T1,T2,TResult>(`
`  Expression<Func<T1,T2,IIndexedSource<TResult>>> query`
`)`

Parameters

query

The query expression to be compiled.

Type Parameters

T1

The type of the first parameter that has to be passed in when executing the delegate returned by the Compile method.

T2

The type of the second parameter that has to be passed in when executing the delegate returned by the Compile method.

TResult

The type of **TResult** in the IIndexedSource<TResult> returned when executing the delegate returned by the Compile method.

Return Value

The delegate to be called to execute the compiled query with particular parameter values.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

CompiledQuery Class
CompiledQuery Members
Overload List

Compile<T,TResult>(Expression<Func<T,IIndexedSource<TResult>>>) Method

The type of the parameter that has to be passed in when executing the delegate returned by the Compile method.

The type of **TResult** in the IIndexedSource<TResult> returned when executing the delegate returned by the Compile method.

The query expression to be compiled.

Compiles the query.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function Compile
    (Of T,TResult)( _
   ByVal query As Expression(Of Func(Of T,IIndexedSource(Of TResult))) _
) As Func(Of T,IIndexedSource(Of TResult))
```

C#

```
public static Func<T,IIndexedSource<TResult>> Compile<T,TResult>(
   Expression<Func<T,IIndexedSource<TResult>>> query
)
```

Parameters

query

The query expression to be compiled.

Type Parameters

T

The type of the parameter that has to be passed in when executing the delegate returned by the Compile method.

TResult

The type of **TResult** in the IIndexedSource<TResult> returned when executing the delegate returned by the Compile method.

Return Value

The delegate to be called to execute the compiled query with particular parameter values.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

CompiledQuery Class
CompiledQuery Members
Overload List

DeletedStateIsAvailableAttribute
Indicates that the properties of an object of a class still return correct property values after the object has been deleted from the collection it belonged to.

Object Model

Syntax

Visual Basic (Declaration)	
`Public NotInheritable Class DeletedStateIsAvailableAttribute` `  Inherits System.Attribute`	

C#

```
public sealed class DeletedStateIsAvailableAttribute : System.Attribute
```

Remarks

This attribute is used with classes of elements of C1.LiveLinq.Collections.IndexedCollection<T> and other collections implementing IObservableSource<T>. It gives you a way of changing the value of IObservableSource<T>.IsDeletedStateAvailable without having to create a full-blown custom implementation of the IObservableSource<T> interface.

Inheritance Hierarchy

System.Object
  System.Attribute
    C1.LiveLinq.DeletedStateIsAvailableAttribute

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

DeletedStateIsAvailableAttribute Members
C1.LiveLinq Namespace

Overview
Indicates that the properties of an object of a class still return correct property values after the object has been deleted from the collection it belonged to.

Object Model

Syntax

Visual Basic (Declaration)

```
Public NotInheritable Class DeletedStateIsAvailableAttribute
   Inherits System.Attribute
```

C#

```
public sealed class DeletedStateIsAvailableAttribute : System.Attribute
```

Remarks

This attribute is used with classes of elements of C1.LiveLinq.Collections.IndexedCollection<T> and other collections implementing IObservableSource<T>. It gives you a way of changing the value of IObservableSource<T>.IsDeletedStateAvailable without having to create a full-blown custom implementation of the IObservableSource<T> interface.

Inheritance Hierarchy

System.Object
  System.Attribute
    C1.LiveLinq.DeletedStateIsAvailableAttribute

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

DeletedStateIsAvailableAttribute Members
C1.LiveLinq Namespace

Members
Properties Methods

The following tables list the members exposed by DeletedStateIsAvailableAttribute.

Public Constructors

	Name	Description
	DeletedStateIsAvailableAttribute Constructor	Initializes a new instance of the

		DeletedStateIsAvailableAttribute class.

Top

Public Properties

	Name	Description
	IsAvailable	**true** if the properties of an object of the class still return correct property values after the object has been deleted from the collection it belonged to.
	TypeId	(Inherited from System.Attribute)

Top

Public Methods

	Name	Description
	Equals	(Inherited from System.Attribute)
	GetHashCode	(Inherited from System.Attribute)
	IsDefaultAttribute	(Inherited from System.Attribute)
	Match	(Inherited from System.Attribute)

Top

See Also

Reference

DeletedStateIsAvailableAttribute Class
C1.LiveLinq Namespace

DeletedStateIsAvailableAttribute Constructor
true if the properties of an object of the class still return correct property values after the object has been deleted from the collection it belonged to.

Initializes a new instance of the DeletedStateIsAvailableAttribute class.

Syntax

Visual Basic (Declaration)

```
Public Function New( _
   ByVal isAvailable As Boolean _
)
```

C#

```
public DeletedStateIsAvailableAttribute(
   bool isAvailable
)
```

Parameters

isAvailable

true if the properties of an object of the class still return correct property values after the object has been deleted from the collection it belonged to.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

DeletedStateIsAvailableAttribute Class
DeletedStateIsAvailableAttribute Members

Properties
For a list of all members of this type, see DeletedStateIsAvailableAttribute members.

Public Properties

	Name	Description
	IsAvailable	**true** if the properties of an object of the class still return correct property

		values after the object has been deleted from the collection it belonged to.
	TypeId	(Inherited from System.Attribute)

Top

See Also

Reference

DeletedStateIsAvailableAttribute Class
C1.LiveLinq Namespace

IsAvailable Property
true if the properties of an object of the class still return correct property values after the object has been deleted from the collection it belonged to.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Property IsAvailable As Boolean
```

C#

```
public bool IsAvailable {get;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

DeletedStateIsAvailableAttribute Class
DeletedStateIsAvailableAttribute Members

Hints
Provides a static (extension) method used as a hint that can be applied to an expression (usually, a property) in a query.

Object Model

Hints

Syntax

Visual Basic (Declaration)
`Public MustInherit NotInheritable Class Hints`
C#
`public static class Hints`

Remarks

A hint does not change the value of the expression it is applied to, it only tells LiveLinq to create and use an index on that property, if possible.

Note: Hints can only be used in indexed queries (with **AsIndexed()** extension method). Using them in live views (with **AsLive()** extension method) will result in an exception.

See How to create indexes.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.Hints

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Hints Members
C1.LiveLinq Namespace

Overview

Provides a static (extension) method used as a hint that can be applied to an expression (usually, a property) in a query.

Object Model

Hints

Syntax

Visual Basic (Declaration)
`Public MustInherit NotInheritable Class Hints`
C#
`public static class Hints`

Remarks

A hint does not change the value of the expression it is applied to, it only tells LiveLinq to create and use an index on that property, if possible.

Note: Hints can only be used in indexed queries (with **AsIndexed()** extension method). Using them in live views (with **AsLive()** extension method) will result in an exception.

See How to create indexes.

Inheritance Hierarchy

System.Object

C1.LiveLinq.Hints

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Hints Members
C1.LiveLinq Namespace

Members
Properties Methods

The following tables list the members exposed by Hints.

Public Properties

	Name	Description
S	DefaultAction	Gets or sets the default action for indexing hints.

Top

Public Methods

	Name	Description
S	Indexed	Overloaded. A hint with specified action.

Top

See Also

Reference

Hints Class
C1.LiveLinq Namespace

Methods
For a list of all members of this type, see Hints members.

Public Methods

	Name	Description
S	Indexed	Overloaded. A hint with specified action.

Top

See Also

Reference

Hints Class
C1.LiveLinq Namespace

Indexed Method
A hint with specified action.

Overload List

Overload	Description
Indexed<T>(T,IndexingHintAction)	A hint with specified action.
Indexed<T>(T)	A hint with default action.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Hints Class
Hints Members

Indexed<T>(T,IndexingHintAction) Method
This can be any type, because hints are applicable to expressions of any type.

The value of the expression the hint is applied to. This value is not actually used by the hint, because the hint is never executed, its role is purely declarative.

The action specified by the hint.

A hint with specified action.

Syntax

Visual Basic (Declaration)	

```
Public Overloads Shared Function Indexed(Of T)( _
   ByVal value As T, _
   ByVal action As IndexingHintAction _
) As T
```

C#

```
public static T Indexed<T>(
   T value,
   IndexingHintAction action
)
```

Parameters

value

The value of the expression the hint is applied to. This value is not actually used by the hint, because the hint is never executed, its role is purely declarative.

action

The action specified by the hint.

Type Parameters

T

This can be any type, because hints are applicable to expressions of any type.

Return Value

Formally, the hint returns the same value that it receives in the parameter. In fact, it is never executed, its role is purely declarative.

Remarks

A hint does not change the value of the expression it is applied to. It only tells LiveLinq query optimizer to create and use an index on that expression, if possible. In fact, hints are removed from the expression before it is executed.

See Also: How to create indexes.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Hints Class
Hints Members
Overload List

Indexed<T>(T) Method

This can be any type, because hints are applicable to expressions of any type.

The value of the expression the hint is applied to. This value is not actually used by the hint, because the hint is never executed, its role is purely declarative.

A hint with default action.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function Indexed(Of T)( _`
`  ByVal value As T _`
`) As T`

C#

`public static T Indexed<T>(`
`  T value`
`)`

Parameters

value

The value of the expression the hint is applied to. This value is not actually used by the hint, because the hint is never executed, its role is purely declarative.

Type Parameters

T

This can be any type, because hints are applicable to expressions of any type.

Return Value

Formally, the hint returns the same value that it receives in the parameter. In fact, it is never executed, its role is purely declarative.

Remarks

Current default action is defined by the value of the static DefaultAction property. By default, it is IndexingHintAction.**Optional**.

Use the other **Indexed** overload if you need a hint with non-default action.

A hint does not change the value of the expression it is applied to. It only tells LiveLinq query optimizer to create and use an index on that expression, if possible. In fact, hints are removed from the expression before it is executed.

See Also: How to create indexes.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Hints Class
Hints Members
Overload List

Properties
For a list of all members of this type, see Hints members.

Public Properties

	Name	Description
S	DefaultAction	Gets or sets the default action for indexing hints.

Top

See Also

Reference

Hints Class
C1.LiveLinq Namespace

DefaultAction Property
Gets or sets the default action for indexing hints.

Syntax

Visual Basic (Declaration)
`Public Shared Property DefaultAction As IndexingHintAction`
C#
`public static IndexingHintAction DefaultAction {get; set;}`

Remarks

The default is IndexingHintAction.**Optional**. This setting is global for the application. If you want to change it, do it at application startup.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

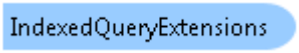
See Also

Reference

Hints Class
Hints Members

IndexedQueryExtensions
Provides a set of static (extension) methods for querying objects that implement C1.LiveLinq.Indexing.IIndexedSource<T>.

Object Model



Syntax

Visual Basic (Declaration)	
`Public MustInherit NotInheritable Class IndexedQueryExtensions`	
C#	
`public static class IndexedQueryExtensions`	

Remarks

The methods in this class provide an implementation of the LINQ query operators for querying data sources that implement C1.LiveLinq.Indexing.IIndexedSource<T>. Those data sources include LiveLinq to Objects, LiveLinq to DataSet and LiveLinq to XML, see How to query collections with LiveLinq.

These implementations of query operators use indexing and other optimization techniques to speed up query execution.

Not all standard LINQ query operators are present here, but it does not mean that they cannot be used in the same query. For the operators that are not present here, standard LINQ implementations are used, because they don't require or don't allow optimization.

Note: Live views are also LiveLinq data sources, but they have their own implementations of query operators defined in the C1.LiveLinq.LiveViews.View<T> class. Live view implementations are heavier, require more resources, so it is not recommended to use live view implementations in cases where you don't need live view functionality (for example, for querying read-only collections), see Live View Performance. If you have a live view, but want to query it using operators from **IndexedQueryExtensions** instead of C1.LiveLinq.LiveViews.View<T>, use AsIndexed<T> to "downgrade" the live view to an C1.LiveLinq.Indexing.IIndexedSource<T>.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.IndexedQueryExtensions

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

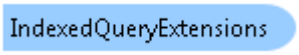
See Also

Reference

IndexedQueryExtensions Members
C1.LiveLinq Namespace

Overview
Provides a set of static (extension) methods for querying objects that implement C1.LiveLinq.Indexing.IIndexedSource<T>.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit NotInheritable Class IndexedQueryExtensions`
C#
`public static class IndexedQueryExtensions`

Remarks

The methods in this class provide an implementation of the LINQ query operators for querying data sources that implement C1.LiveLinq.Indexing.IIndexedSource<T>. Those data sources include LiveLinq to Objects, LiveLinq to DataSet and LiveLinq to XML, see How to query collections with LiveLinq.

These implementations of query operators use indexing and other optimization techniques to speed up query execution.

Not all standard LINQ query operators are present here, but it does not mean that they cannot be used in the same query. For the operators that are not present here, standard LINQ implementations are used, because they don't require or don't allow optimization.

Note: Live views are also LiveLinq data sources, but they have their own implementations of query operators defined in the C1.LiveLinq.LiveViews.View<T> class. Live view implementations are heavier, require more resources, so it is not recommended to use live view implementations in cases where you don't need live view functionality (for example, for querying read-only collections), see Live View Performance. If you have a live view, but want to query it using operators from **IndexedQueryExtensions** instead of C1.LiveLinq.LiveViews.View<T>, use AsIndexed<T> to "downgrade" the live view to an C1.LiveLinq.Indexing.IIndexedSource<T>.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.IndexedQueryExtensions

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedQueryExtensions Members
C1.LiveLinq Namespace

Members
Methods

The following tables list the members exposed by IndexedQueryExtensions.

Public Methods

	Name	Description
S	AsIndexed<T>	Returns the input typed as C1.LiveLinq.Indexing.IIndexedSource<T>.
S	GroupBy	Overloaded. Groups the elements of a collection according to a specified key selector function and creates a result value from each group and its key. The elements of each group are projected by using a specified function.
S	GroupJoin<TOuter,TInner,TKey,TResult>	Correlates the elements of two collections based on equality of keys and groups the results.
S	Join	Overloaded. Correlates the elements of two

		collections based on matching keys.
S	OrderBy<T,TKey>	Sorts the elements of a collection in ascending order.
S	OrderByDescending<T,TKey>	Sorts the elements of a collection in descending order.
S	Select<TSource,TResult>	Projects each element of a collection into a new form.
S	SelectMany	Overloaded. Projects each element of a collection to a sequence of collections, flattens the resulting collections into one collection, and invokes a result selector function on each element therein.
S	ToIndexed	Overloaded. Creates an C1.LiveLinq.Indexing.IIndexedSource<T> based on the specified IObservableSource<T> collection.
S	Where	Overloaded. Filters the source collection based on a predicate.

Top

See Also

Reference

IndexedQueryExtensions Class
C1.LiveLinq Namespace

Methods
For a list of all members of this type, see IndexedQueryExtensions members.

Public Methods

	Name	Description

S	AsIndexed<T>	Returns the input typed as C1.LiveLinq.Indexing.IIndexedSource<T>.
S	GroupBy	Overloaded. Groups the elements of a collection according to a specified key selector function and creates a result value from each group and its key. The elements of each group are projected by using a specified function.
S	GroupJoin<TOuter,TInner,TKey,TResult>	Correlates the elements of two collections based on equality of keys and groups the results.
S	Join	Overloaded. Correlates the elements of two collections based on matching keys.
S	OrderBy<T,TKey>	Sorts the elements of a collection in ascending order.
S	OrderByDescending<T,TKey>	Sorts the elements of a collection in descending order.
S	Select<TSource,TResult>	Projects each element of a collection into a new form.
S	SelectMany	Overloaded. Projects each element of a collection to a sequence of collections, flattens the resulting collections into one collection, and invokes a result selector function on each element therein.
S	ToIndexed	Overloaded. Creates an C1.LiveLinq.Indexing.IIndexedSource<T> based on the specified IObservableSource<T> collection.
S	Where	Overloaded. Filters the source collection based on

		a predicate.

Top

See Also

Reference

IndexedQueryExtensions Class
C1.LiveLinq Namespace

AsIndexed<T> Method
The type of the elements of *source*.

The collection to type as C1.LiveLinq.Indexing.IIndexedSource<T>.

Returns the input typed as C1.LiveLinq.Indexing.IIndexedSource<T>.

Syntax

Visual Basic (Declaration)

```
Public Shared Function AsIndexed(Of T)( _
   ByVal source As IIndexedSource(Of T) _
) As IIndexedSource(Of T)
```

C#

```
public static IIndexedSource<T> AsIndexed<T>(
   IIndexedSource<T> source
)
```

Parameters

source

The collection to type as C1.LiveLinq.Indexing.IIndexedSource<T>.

Type Parameters

T

The type of the elements of *source*.

Return Value

The input collection typed as C1.LiveLinq.Indexing.IIndexedSource<T>.

Remarks

This method has no effect other than to change the compile-time type of *source* from a type that implements C1.LiveLinq.Indexing.IIndexedSource<T> to C1.LiveLinq.Indexing.IIndexedSource<T> itself. It is used to choose between query implementations when a collection implements C1.LiveLinq.Indexing.IIndexedSource<T> but also has a different set of public query methods available.

The main scenario is when you want to perform queries on live views without creating new live views.

Live views have their own implementations of query operators defined in the C1.LiveLinq.LiveViews.View<T> class. Live view implementations are heavier, require more resources, so it is not recommended to use live view implementations in cases where you don't need live view functionality (for example, for querying read-only collections), see Live View Performance. If you have a live view, but want to query it using operators from IndexedQueryExtensions instead of C1.LiveLinq.LiveViews.View<T>, use this method to "downgrade" the live view to an C1.LiveLinq.Indexing.IIndexedSource<T>. This method does not cause the live view to lose its "live" functionality, it simply returns the live view's implementation of the C1.LiveLinq.Indexing.IIndexedSource<T> interface.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedQueryExtensions Class
IndexedQueryExtensions Members

GroupBy Method
Groups the elements of a collection according to a specified key selector function and creates a result value from each group and its key. The elements of each group are projected by using a specified function.

Overload List

Overload	Description
GroupBy<TSource,TKey,TElement,TResult>(IIndexedSource<TSource>,Expression<Func<TSource,TKey>>,Expression<Func<TSource,TElement>>,Expression<Func<TKey,IEnumerable<TElement>,TResult>>)	Groups the elements of a collection according to a specified key selector function and creates a result value from each group and

	its key. The elements of each group are projected by using a specified function.
GroupBy<TSource,TKey,TElement>(IIndexedSource<TSource>,Expression<Func<TSource,TKey>>,Expression<Func<TSource,TElement>>)	Groups the elements of a collection according to a specified key

	selector function and projects the elements for each group by using a specified function.
GroupBy<TSource,TKey,TResult>(IIndexedSource<TSource>,Expression<Func<TSource,TKey>>,Expression<Func<TKey,IEnumerable<TSource>,TResult>>)	Groups the elements of a collection according to a speci

	fied key selector function and creates a result value from each group and its key.
GroupBy<TSource,TKey>(IIndexedSource<TSource>,Expression<Func<TSource,TKey>>)	Groups the elements of a collection according to a speci

	fied key selec tor funct ion.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedQueryExtensions Class
IndexedQueryExtensions Members

GroupBy<TSource,TKey,TElement,TResult>(IIndexedSource<TSource>,Expression<Func<TSource,TKey>>, Expression<Func<TSource,TElement>>,Expression<Func<TKey,IEnumerable<TElement>,TResult>>) Method
The type of the elements of *source*.

The type of the key returned by *keySelector*.

The type of the elements in the IGrouping.

The type of the result value returned by *resultSelector*

An C1.LiveLinq.Indexing.IIndexedSource<T> whose elements to group

A function to extract the key for each element.

A function to map each source element to an element in the IGrouping.

A function to create a result value from each group.

Groups the elements of a collection according to a specified key selector function and creates a result value from each group and its key. The elements of each group are projected by using a specified function.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function GroupBy
    (Of TSource,TKey,TElement,TResult)( _
   ByVal source As IIndexedSource(Of TSource), _
   ByVal keySelector As Expression(Of Func(Of TSource,TKey)), _
   ByVal elementSelector As Expression(Of Func(Of TSource,TElement)), _
   ByVal resultSelector As Expression(Of Func(Of TKey,IEnumerable(Of
TElement),TResult)) _
) As IIndexedSource(Of TResult)
```

C#

```
public static IIndexedSource<TResult> GroupBy<TSource,TKey,TElement,TResult>(
   IIndexedSource<TSource> source,
   Expression<Func<TSource,TKey>> keySelector,
   Expression<Func<TSource,TElement>> elementSelector,
   Expression<Func<TKey,IEnumerable<TElement>,TResult>> resultSelector
)
```

Parameters

source

An C1.LiveLinq.Indexing.IIndexedSource<T> whose elements to group

keySelector

A function to extract the key for each element.

elementSelector

A function to map each source element to an element in the IGrouping.

resultSelector

A function to create a result value from each group.

Type Parameters

TSource

The type of the elements of *source*.

TKey

The type of the key returned by *keySelector*.

TElement

The type of the elements in the IGrouping.

TResult

The type of the result value returned by *resultSelector*

Return Value

A collection of elements of type *TResult* where each element represents a projection over a group and its key.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedQueryExtensions Class
IndexedQueryExtensions Members
Overload List

GroupBy<TSource,TKey,TElement>(IIndexedSource<TSource>,Expression<Func<TSource,TKey>>,Expression<Func<TSource,TElement>>) Method
The type of the elements of *source*.

The type of the key returned by *keySelector*.

The type of the elements in the IGrouping.

An C1.LiveLinq.Indexing.IIndexedSource<T> whose elements to group

A function to extract the key for each element.

A function to map each source element to an element in the IGrouping.

Groups the elements of a collection according to a specified key selector function and projects the elements for each group by using a specified function.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function GroupBy
    (Of TSource,TKey,TElement)( _
   ByVal source As IIndexedSource(Of TSource), _
   ByVal keySelector As Expression(Of Func(Of TSource,TKey)), _
   ByVal elementSelector As Expression(Of Func(Of TSource,TElement)) _
) As IIndexedSource(Of IGrouping(Of TKey,TElement))
```

C#

```
public static IIndexedSource<IGrouping<TKey,TElement>>
GroupBy<TSource,TKey,TElement>(
   IIndexedSource<TSource> source,
   Expression<Func<TSource,TKey>> keySelector,
   Expression<Func<TSource,TElement>> elementSelector
)
```

Parameters

source

An C1.LiveLinq.Indexing.IIndexedSource<T> whose elements to group

keySelector

A function to extract the key for each element.

elementSelector

A function to map each source element to an element in the IGrouping.

Type Parameters

TSource

The type of the elements of *source*.

TKey

The type of the key returned by *keySelector*.

TElement

The type of the elements in the IGrouping.

Return Value

A collection of IGrouping objects each containing a collection of objects of type *TElement* and a key.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedQueryExtensions Class
IndexedQueryExtensions Members
Overload List

GroupBy<TSource,TKey,TResult>(IIndexedSource<TSource>,Expression<Func<TSource,TKey>>,Expression<Func<TKey,IEnumerable<TSource>,TResult>>) Method
The type of the elements of *source*.

The type of the key returned by *keySelector*.

The type of the result value returned by *resultSelector*

An C1.LiveLinq.Indexing.IIndexedSource<T> whose elements to group

A function to extract the key for each element.

A function to create a result value from each group.

Groups the elements of a collection according to a specified key selector function and creates a result value from each group and its key.

Syntax

Visual Basic (Declaration)	
`Public Overloads Shared Function GroupBy`	

```
    (Of TSource,TKey,TResult)( _
   ByVal source As IIndexedSource(Of TSource), _
   ByVal keySelector As Expression(Of Func(Of TSource,TKey)), _
   ByVal resultSelector As Expression(Of Func(Of TKey,IEnumerable(Of
TSource),TResult)) _
) As IIndexedSource(Of TResult)
```

C#

```
public static IIndexedSource<TResult> GroupBy<TSource,TKey,TResult>(
   IIndexedSource<TSource> source,
   Expression<Func<TSource,TKey>> keySelector,
   Expression<Func<TKey,IEnumerable<TSource>,TResult>> resultSelector
)
```

Parameters

source

An C1.LiveLinq.Indexing.IIndexedSource<T> whose elements to group

keySelector

A function to extract the key for each element.

resultSelector

A function to create a result value from each group.

Type Parameters

TSource

The type of the elements of *source*.

TKey

The type of the key returned by *keySelector*.

TResult

The type of the result value returned by *resultSelector*

Return Value

A collection of elements of type *TResult* where each element represents a projection over a group and its key.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedQueryExtensions Class
IndexedQueryExtensions Members
Overload List

GroupBy<TSource,TKey>(IIndexedSource<TSource>,Expression<Func<TSource,TKey>>) Method

The type of the elements of *source*.

The type of the key returned by *keySelector*.

An C1.LiveLinq.Indexing.IIndexedSource<T> whose elements to group

A function to extract the key for each element.

Groups the elements of a collection according to a specified key selector function.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function GroupBy
    (Of TSource,TKey)( _
   ByVal source As IIndexedSource(Of TSource), _
   ByVal keySelector As Expression(Of Func(Of TSource,TKey)) _
) As IIndexedSource(Of IGrouping(Of TKey,TSource))
```

C#

```
public static IIndexedSource<IGrouping<TKey,TSource>> GroupBy<TSource,TKey>(
   IIndexedSource<TSource> source,
   Expression<Func<TSource,TKey>> keySelector
)
```

Parameters

source

An C1.LiveLinq.Indexing.IIndexedSource<T> whose elements to group

keySelector

A function to extract the key for each element.

Type Parameters

TSource

The type of the elements of *source*.

TKey

The type of the key returned by *keySelector*.

Return Value

A collection of IGrouping objects each containing a sequence of objects and a key.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedQueryExtensions Class
IndexedQueryExtensions Members
Overload List

GroupJoin<TOuter,TInner,TKey,TResult> Method
The type of the elements of the first collection.

The type of the elements of the second collection.

The type of the keys returned by the key selector functions.

The type of the result elements.

The first collection to join.

The collection to join to the first collection.

A function to extract the join key from each element of the first collection.

A function to extract the join key from each element of the second collection.

A function to create a result element from an element from the first collection and a collection of matching elements from the second collection.

Correlates the elements of two collections based on equality of keys and groups the results.

Syntax

Visual Basic (Declaration)

```
Public Shared Function GroupJoin
    (Of TOuter,TInner,TKey,TResult)( _
   ByVal outer As IIndexedSource(Of TOuter), _
   ByVal inner As IEnumerable(Of TInner), _
   ByVal outerKeySelector As Expression(Of Func(Of TOuter,TKey)), _
   ByVal innerKeySelector As Expression(Of Func(Of TInner,TKey)), _
   ByVal resultSelector As Expression(Of Func(Of TOuter,IEnumerable(Of
TInner),TResult)) _
) As IIndexedSource(Of TResult)
```

C#

```
public static IIndexedSource<TResult> GroupJoin<TOuter,TInner,TKey,TResult>(
   IIndexedSource<TOuter> outer,
   IEnumerable<TInner> inner,
   Expression<Func<TOuter,TKey>> outerKeySelector,
   Expression<Func<TInner,TKey>> innerKeySelector,
   Expression<Func<TOuter,IEnumerable<TInner>,TResult>> resultSelector
)
```

Parameters

outer

The first collection to join.

inner

The collection to join to the first collection.

outerKeySelector

A function to extract the join key from each element of the first collection.

innerKeySelector

A function to extract the join key from each element of the second collection.

resultSelector

A function to create a result element from an element from the first collection and a collection of matching elements from the second collection.

Type Parameters

TOuter

The type of the elements of the first collection.

TInner

The type of the elements of the second collection.

TKey

The type of the keys returned by the key selector functions.

TResult

The type of the result elements.

Return Value

An IIndexedSource<TResult> that contains elements of type *TResult* that are obtained by performing a grouped join on two collections.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedQueryExtensions Class
IndexedQueryExtensions Members

Join Method

Correlates the elements of two collections based on matching keys.

Overload List

Overload	Description
Join<TOuter,TInner,TKey,TResult>(IIndexedSource<TOuter>,IEnumerable<TInner>,Expression<Func<TOuter,TKey>>,Expression<Func<TInner,TKey>>,Expression<Func<TOuter,TInner,TResult>>)	Correlates the elements of two collections based on matching keys.
Join<TOuter,TInner,TKey,TResult>(IEnumerable<TOuter>,IIndexedSource<TInner>,Expression<Func<TOuter,TKey>>,Expression<Func<TInner,TKey>>,Expression<Func<TOuter,TInner,TResult>>)	Correlates the elements of two colle

	ctions based on matching keys.
Join<TOuter,TInner,TKey,TResult>(IIndexedSource<TOuter>,IIndexedSource<TInner>,Expression<Func<TOuter,TKey>>,Expression<Func<TInner,TKey>>,Expression<Func<TOuter,TInner,TResult>>)	Correlates the elements of two collections based on matching keys.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedQueryExtensions Class
IndexedQueryExtensions Members

Join<TOuter,TInner,TKey,TResult>(IIndexedSource<TOuter>,IEnumerable<TInner>,Expression<Func<TOuter,TKey>>,Expression<Func<TInner,TKey>>,Expression<Func<TOuter,TInner,TResult>>) Method

The type of the elements of the first collection.

The type of the elements of the second collection.

The type of the keys returned by the key selector functions.

The type of the result elements.

The first collection to join.

The second collection to join.

A function to extract the join key from each element of the first collection.

A function to extract the join key from each element of the second collection.

A function to create a result element from two matching elements.

Correlates the elements of two collections based on matching keys.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function Join
    (Of TOuter,TInner,TKey,TResult)( _
   ByVal outer As IIndexedSource(Of TOuter), _
   ByVal inner As IEnumerable(Of TInner), _
   ByVal outerKeySelector As Expression(Of Func(Of TOuter,TKey)), _
   ByVal innerKeySelector As Expression(Of Func(Of TInner,TKey)), _
   ByVal resultSelector As Expression(Of Func(Of TOuter,TInner,TResult)) _
) As IIndexedSource(Of TResult)
```

C#

```
public static IIndexedSource<TResult> Join<TOuter,TInner,TKey,TResult>(
   IIndexedSource<TOuter> outer,
   IEnumerable<TInner> inner,
   Expression<Func<TOuter,TKey>> outerKeySelector,
   Expression<Func<TInner,TKey>> innerKeySelector,
```

```
    Expression<Func<TOuter,TInner,TResult>> resultSelector
)
```

Parameters

outer

The first collection to join.

inner

The second collection to join.

outerKeySelector

A function to extract the join key from each element of the first collection.

innerKeySelector

A function to extract the join key from each element of the second collection.

resultSelector

A function to create a result element from two matching elements.

Type Parameters

TOuter

The type of the elements of the first collection.

TInner

The type of the elements of the second collection.

TKey

The type of the keys returned by the key selector functions.

TResult

The type of the result elements.

Return Value

An IIndexedSource<TResult> that has elements of type *TResult* that are obtained by performing an inner join on two collections.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedQueryExtensions Class
IndexedQueryExtensions Members
Overload List

Join<TOuter,TInner,TKey,TResult>(IEnumerable<TOuter>,IIndexedSource<TInner>,Expression<Func<TOuter,TKey>>,Expression<Func<TInner,TKey>>,Expression<Func<TOuter,TInner,TResult>>) Method

The type of the elements of the first collection.

The type of the elements of the second collection.

The type of the keys returned by the key selector functions.

The type of the result elements.

The first collection to join.

The second collection to join.

A function to extract the join key from each element of the first collection.

A function to extract the join key from each element of the second collection.

A function to create a result element from two matching elements.

Correlates the elements of two collections based on matching keys.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function Join
    (Of TOuter,TInner,TKey,TResult)( _
   ByVal outer As IEnumerable(Of TOuter), _
   ByVal inner As IIndexedSource(Of TInner), _
   ByVal outerKeySelector As Expression(Of Func(Of TOuter,TKey)), _
```

```
   ByVal innerKeySelector As Expression(Of Func(Of TInner,TKey)), _
   ByVal resultSelector As Expression(Of Func(Of TOuter,TInner,TResult)) _
) As IIndexedSource(Of TResult)
```

C#

```
public static IIndexedSource<TResult> Join<TOuter,TInner,TKey,TResult>(
   IEnumerable<TOuter> outer,
   IIndexedSource<TInner> inner,
   Expression<Func<TOuter,TKey>> outerKeySelector,
   Expression<Func<TInner,TKey>> innerKeySelector,
   Expression<Func<TOuter,TInner,TResult>> resultSelector
)
```

Parameters

outer

The first collection to join.

inner

The second collection to join.

outerKeySelector

A function to extract the join key from each element of the first collection.

innerKeySelector

A function to extract the join key from each element of the second collection.

resultSelector

A function to create a result element from two matching elements.

Type Parameters

TOuter

The type of the elements of the first collection.

TInner

The type of the elements of the second collection.

TKey

The type of the keys returned by the key selector functions.

TResult

The type of the result elements.

Return Value

An IIndexedSource<TResult> that has elements of type *TResult* that are obtained by performing an inner join on two collections.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedQueryExtensions Class
IndexedQueryExtensions Members
Overload List

Join<TOuter,TInner,TKey,TResult>(IIndexedSource<TOuter>,IIndexedSource<TInner>,Expression<Func<TOuter,TKey>>,Expression<Func<TInner,TKey>>,Expression<Func<TOuter,TInner,TResult>>) Method

The type of the elements of the first collection.

The type of the elements of the second collection.

The type of the keys returned by the key selector functions.

The type of the result elements.

The first collection to join.

The second collection to join.

A function to extract the join key from each element of the first collection.

A function to extract the join key from each element of the second collection.

A function to create a result element from two matching elements.

Correlates the elements of two collections based on matching keys.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function Join
    (Of TOuter,TInner,TKey,TResult)( _
   ByVal outer As IIndexedSource(Of TOuter), _
   ByVal inner As IIndexedSource(Of TInner), _
   ByVal outerKeySelector As Expression(Of Func(Of TOuter,TKey)), _
   ByVal innerKeySelector As Expression(Of Func(Of TInner,TKey)), _
   ByVal resultSelector As Expression(Of Func(Of TOuter,TInner,TResult)) _
) As IIndexedSource(Of TResult)
```

C#

```
public static IIndexedSource<TResult> Join<TOuter,TInner,TKey,TResult>(
   IIndexedSource<TOuter> outer,
   IIndexedSource<TInner> inner,
   Expression<Func<TOuter,TKey>> outerKeySelector,
   Expression<Func<TInner,TKey>> innerKeySelector,
   Expression<Func<TOuter,TInner,TResult>> resultSelector
)
```

Parameters

outer

The first collection to join.

inner

The second collection to join.

outerKeySelector

A function to extract the join key from each element of the first collection.

innerKeySelector

A function to extract the join key from each element of the second collection.

resultSelector

A function to create a result element from two matching elements.

Type Parameters

TOuter

The type of the elements of the first collection.

TInner

The type of the elements of the second collection.

TKey

The type of the keys returned by the key selector functions.

TResult

The type of the result elements.

Return Value

An IIndexedSource<TResult> that has elements of type *TResult* that are obtained by performing an inner join on two collections.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedQueryExtensions Class
IndexedQueryExtensions Members
Overload List

OrderBy<T,TKey> Method
The type of the elements of *source*.

The type of the key returned by *keySelector*.

A collection of values to order.

A function to extract a key from an element.

Sorts the elements of a collection in ascending order.

Syntax

Visual Basic (Declaration)

```
Public Shared Function OrderBy
    (Of T,TKey)( _
   ByVal source As IIndexedSource(Of T), _
   ByVal keySelector As Expression(Of Func(Of T,TKey)) _
) As Ordering(Of T)
```

C#

```
public static Ordering<T> OrderBy<T,TKey>(
   IIndexedSource<T> source,
   Expression<Func<T,TKey>> keySelector
)
```

Parameters

source

A collection of values to order.

keySelector

A function to extract a key from an element.

Type Parameters

T

The type of the elements of *source*.

TKey

The type of the key returned by *keySelector*.

Return Value

An C1.LiveLinq.Indexing.IIndexedSource<T> whose elements are sorted according to a key.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedQueryExtensions Class
IndexedQueryExtensions Members

OrderByDescending<T,TKey> Method
The type of the elements of *source*.

The type of the key returned by *keySelector*.

A collection of values to order.

A function to extract a key from an element.

Sorts the elements of a collection in descending order.

Syntax

Visual Basic (Declaration)

```
Public Shared Function OrderByDescending
    (Of T,TKey)( _
   ByVal source As IIndexedSource(Of T), _
   ByVal keySelector As Expression(Of Func(Of T,TKey)) _
) As Ordering(Of T)
```

C#

```
public static Ordering<T> OrderByDescending<T,TKey>(
   IIndexedSource<T> source,
   Expression<Func<T,TKey>> keySelector
)
```

Parameters

source

A collection of values to order.

keySelector

A function to extract a key from an element.

Type Parameters

T

The type of the elements of *source*.

TKey

The type of the key returned by *keySelector*.

Return Value

An C1.LiveLinq.Indexing.IIndexedSource<T> whose elements are sorted in descending order according to a key.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedQueryExtensions Class
IndexedQueryExtensions Members

Select<TSource,TResult> Method
The type of the elements of *source*.

The type of the value returned by *selector*.

An IIndexedSource<TSource> collection of values to invoke a transform function on.

A transform function to apply to each element.

Projects each element of a collection into a new form.

Syntax

Visual Basic (Declaration)

```
Public Shared Function Select
    (Of TSource,TResult)( _
   ByVal source As IIndexedSource(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,TResult)) _
) As IIndexedSource(Of TResult)
```

C#

```
public static IIndexedSource<TResult> Select<TSource,TResult>(
   IIndexedSource<TSource> source,
   Expression<Func<TSource,TResult>> selector
)
```

Parameters

source

An IIndexedSource<TSource> collection of values to invoke a transform function on.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

TResult

The type of the value returned by *selector*.

Return Value

An IIndexedSource<TResult> whose elements are the result of invoking the transform function on each element of *source*.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedQueryExtensions Class
IndexedQueryExtensions Members

SelectMany Method
Projects each element of a collection to a sequence of collections, flattens the resulting collections into one collection, and invokes a result selector function on each element therein.

Overload List

Overload	Description
SelectMany<TSource,TCollection,TResult>(IIndexedSource<TSource>,Expression<Func<TSource,IEnumerable<TCollection>>>,Expression<Func<TSource,TCollection,TResult>>)	Projects each element of a collection to a sequence of collections, flattens the resulting collections

	into one collection, and invokes a result selector function on each element therein.
SelectMany<TSource,TResult>(IIndexedSource<TSource>,Expression<Func<TSource,IEnumerable<TResult>>>)	Projects each element of a collection to a sequence of collections and flatte

	ns the resulting collections into one collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedQueryExtensions Class
IndexedQueryExtensions Members

SelectMany<TSource,TCollection,TResult>(IIndexedSource<TSource>,Expression<Func<TSource,IEnumerable<TCollection>>>,Expression<Func<TSource,TCollection,TResult>>) Method
The type of the elements of *source*.

The type of the intermediate elements collected by *collectionSelector*.

The type of the elements of the resulting collection.

A collection of values to project.

A transform function to apply to each element of the input collection.

A transform function to apply to each element of the intermediate sequence.

Projects each element of a collection to a sequence of collections, flattens the resulting collections into one collection, and invokes a result selector function on each element therein.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function SelectMany
    (Of TSource,TCollection,TResult)( _
   ByVal source As IIndexedSource(Of TSource), _
   ByVal collectionSelector As Expression(Of Func(Of TSource,IEnumerable(Of
TCollection))), _
   ByVal resultSelector As Expression(Of Func(Of TSource,TCollection,TResult)) _
) As IIndexedSource(Of TResult)
```

C#

```
public static IIndexedSource<TResult> SelectMany<TSource,TCollection,TResult>(
   IIndexedSource<TSource> source,
   Expression<Func<TSource,IEnumerable<TCollection>>> collectionSelector,
   Expression<Func<TSource,TCollection,TResult>> resultSelector
)
```

Parameters

source

A collection of values to project.

collectionSelector

A transform function to apply to each element of the input collection.

resultSelector

A transform function to apply to each element of the intermediate sequence.

Type Parameters

TSource

The type of the elements of *source*.

TCollection

The type of the intermediate elements collected by *collectionSelector*.

TResult

The type of the elements of the resulting collection.

Return Value

An IIndexedSource<TResult> whose elements are the result of invoking the one-to-many transform function *collectionSelector* on each element of *source* and then mapping each of those sequence elements and their corresponding source element to a result element.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedQueryExtensions Class
IndexedQueryExtensions Members
Overload List

SelectMany<TSource,TResult>(IIndexedSource<TSource>,Expression<Func<TSource,IEnumerable<TResult>>>) Method

The type of the elements of *source*.

The type of the elements of the sequence returned by *selector*.

A collection of values to project.

A transform function to apply to each element.

Projects each element of a collection to a sequence of collections and flattens the resulting collections into one collection.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function SelectMany
    (Of TSource,TResult)( _
   ByVal source As IIndexedSource(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,IEnumerable(Of TResult))) _
) As IIndexedSource(Of TResult)
```

C#

```
public static IIndexedSource<TResult> SelectMany<TSource,TResult>(
   IIndexedSource<TSource> source,
   Expression<Func<TSource,IEnumerable<TResult>>> selector
)
```

Parameters

source

A collection of values to project.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

TResult

The type of the elements of the sequence returned by *selector*.

Return Value

An IIndexedSource<TResult> whose elements are the result of invoking the one-to-many transform function on each element of the source collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedQueryExtensions Class
IndexedQueryExtensions Members
Overload List

ToIndexed Method

Creates an C1.LiveLinq.Indexing.IIndexedSource<T> based on the specified IObservableSource<T> collection.

Overload List

Overload	Description
ToIndexed<T>(IObservableSource<T>)	Creates an C1.LiveLinq.Indexing.IIndexedSource<T> based on the specified IObservableSource<T> collection.
ToIndexed<T>(IBindingList)	Creates an C1.LiveLinq.Collections.IndexedCollection<T> based on the specified System.ComponentModel.IBindingList data source.
ToIndexed<T>(BindingList<T>)	A typed specialization of the ToIndexed<T>(IBindingList) method.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedQueryExtensions Class
IndexedQueryExtensions Members

ToIndexed<T>(IObservableSource<T>) Method

The type of the elements in the collection.

An IObservableSource<T> collection to base an C1.LiveLinq.Indexing.IIndexedSource<T> on.

Creates an C1.LiveLinq.Indexing.IIndexedSource<T> based on the specified IObservableSource<T> collection.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function ToIndexed(Of T)( _
   ByVal source As IObservableSource(Of T) _
) As IIndexedSource(Of T)
```

C#

```
public static IIndexedSource<T> ToIndexed<T>(
   IObservableSource<T> source
)
```

Parameters

source

An IObservableSource<T> collection to base an C1.LiveLinq.Indexing.IIndexedSource<T> on.

Type Parameters

T

The type of the elements in the collection.

Return Value

An C1.LiveLinq.Indexing.IIndexedSource<T> that contains the same elements as the IObservableSource<T> collection and enables indexing of that collection.

Remarks

Use this method to index and query your collection if that collection is your own custom implementation of the IObservableSource<T> interface.

Elements of the source collection aren't duplicated or copied to a new collection. This method just wraps the original collection in an C1.LiveLinq.Indexing.IIndexedSource<T>, enabling its indexing by using the change notification mechanism of IObservableSource<T>.

Note: Indexes created on the resulting C1.LiveLinq.Indexing.IIndexedSource<T> are owned by it and not by the original collection. Every **ToIndexed()** call creates a separate object that has its own separate indexes. Avoid calling **ToIndexed()** repeatedly for the same collection because it can increase the cost of maintaining indexes.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedQueryExtensions Class
IndexedQueryExtensions Members
Overload List

ToIndexed<T>(IBindingList) Method
The type of the elements in the collection.

An System.ComponentModel.IBindingList data source to represent as an C1.LiveLinq.Collections.IndexedCollection<T>.

Creates an C1.LiveLinq.Collections.IndexedCollection<T> based on the specified System.ComponentModel.IBindingList data source.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function ToIndexed(Of T)( _
   ByVal source As IBindingList _
) As IndexedCollection(Of T)
```

C#

```
public static IndexedCollection<T> ToIndexed<T>(
   IBindingList source
)
```

Parameters

source

An System.ComponentModel.IBindingList data source to represent as an C1.LiveLinq.Collections.IndexedCollection<T>.

Type Parameters

T

The type of the elements in the collection.

Return Value

An C1.LiveLinq.Collections.IndexedCollection<T> that contains the same elements as the System.ComponentModel.IBindingList and enables indexing of that data source.

Remarks

Use this method to index and query your existing data sources. The only requirements for the data source is that it implements the standard data binding interface System.ComponentModel.IBindingList.

Note: Indexes created on the resulting C1.LiveLinq.Collections.IndexedCollection<T> are owned by it and not by the original data source. Every **ToIndexed()** call creates a separate object that has its own separate indexes. Avoid calling **ToIndexed()** repeatedly for the same collection because it can increase the cost of maintaining indexes.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedQueryExtensions Class
IndexedQueryExtensions Members
Overload List

ToIndexed<T>(BindingList<T>) Method
The type of the elements in the collection.

A BindingList data source to represent as an C1.LiveLinq.Collections.IndexedCollection<T>.

A typed specialization of the ToIndexed<T>(IBindingList) method.

Syntax

Visual Basic (Declaration)	
`Public Overloads Shared Function ToIndexed(Of T)( _`	

```
   ByVal source As BindingList(Of T) _
) As IndexedCollection(Of T)
```

C#

```
public static IndexedCollection<T> ToIndexed<T>(
   BindingList<T> source
)
```

Parameters

source

A BindingList data source to represent as an C1.LiveLinq.Collections.IndexedCollection<T>.

Type Parameters

T

The type of the elements in the collection.

Return Value

An C1.LiveLinq.Indexing.IIndexedSource<T> that contains the same elements as the BindingList and enables indexing of that data source.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedQueryExtensions Class
IndexedQueryExtensions Members
Overload List

Where Method
Filters the source collection based on a predicate.

Overload List

Overload	Description
Where<T>(IIndexedSource<T>,Expression<Func<T,Boolean>>)	Filters the source collection based on a predicate.
Where<T>(IIndexedSource<T>,Expression<Func<T,Boolean>>,Boolean)	Filters the source collection based on a predicate, preserving the order of the source collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedQueryExtensions Class
IndexedQueryExtensions Members

Where<T>(IIndexedSource<T>,Expression<Func<T,Boolean>>) Method
The type of the elements of *source*.

An C1.LiveLinq.Indexing.IIndexedSource<T> to filter.

A function to test each element for a condition.

Filters the source collection based on a predicate.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function Where(Of T)( _
   ByVal source As IIndexedSource(Of T), _
   ByVal predicate As Expression(Of Func(Of T,Boolean)) _
```

```
) As IIndexedSource(Of T)
```

C#

```
public static IIndexedSource<T> Where<T>(
   IIndexedSource<T> source,
   Expression<Func<T,bool>> predicate
)
```

Parameters

source

An C1.LiveLinq.Indexing.IIndexedSource<T> to filter.

predicate

A function to test each element for a condition.

Type Parameters

T

The type of the elements of *source*.

Return Value

An C1.LiveLinq.Indexing.IIndexedSource<T> that contains elements from the input collection that satisfy the condition.

Remarks

If an index is used to optimized performance of this operation, the resulting collection may not be in the same order as the source collection. If you need to preserve the order, use the other overload of the **Where** operator. It will still be optimized, albeit to a lesser degree.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedQueryExtensions Class
IndexedQueryExtensions Members
Overload List

Where<T>(IIndexedSource<T>,Expression<Func<T,Boolean>>,Boolean) Method
The type of the elements of *source*.

An C1.LiveLinq.Indexing.IIndexedSource<T> to filter.

A function to test each element for a condition.

Specifies whether the source order must be preserved in the result.

Filters the source collection based on a predicate, preserving the order of the source collection.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function Where(Of T)( _
   ByVal source As IIndexedSource(Of T), _
   ByVal predicate As Expression(Of Func(Of T,Boolean)), _
   ByVal preserveOriginalOrder As Boolean _
) As IIndexedSource(Of T)
```

C#

```
public static IIndexedSource<T> Where<T>(
   IIndexedSource<T> source,
   Expression<Func<T,bool>> predicate,
   bool preserveOriginalOrder
)
```

Parameters

source

An C1.LiveLinq.Indexing.IIndexedSource<T> to filter.

predicate

A function to test each element for a condition.

preserveOriginalOrder

Specifies whether the source order must be preserved in the result.

Type Parameters

T

The type of the elements of *source*.

Return Value

An C1.LiveLinq.Indexing.IIndexedSource<T> that contains elements from the input collection that satisfy the condition.

Remarks

Preserving the order can lessen the effect of performance optimization using an index. Use this overload only if preserving the order in this operation is essential.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

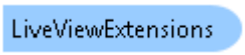
See Also

Reference

IndexedQueryExtensions Class
IndexedQueryExtensions Members
Overload List

LiveViewExtensions
Provides a set of static (extension) methods used in queries to define live views.

Object Model



Syntax

Visual Basic (Declaration)	
`Public MustInherit NotInheritable Class LiveViewExtensions`	

C#

```
public static class LiveViewExtensions
```

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViewExtensions

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

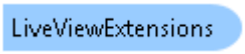
See Also

Reference

LiveViewExtensions Members
C1.LiveLinq Namespace

Overview
Provides a set of static (extension) methods used in queries to define live views.

Object Model



Syntax

Visual Basic (Declaration)

```
Public MustInherit NotInheritable Class LiveViewExtensions
```

C#

```
public static class LiveViewExtensions
```

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViewExtensions

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Members
C1.LiveLinq Namespace

Members
Methods

The following tables list the members exposed by LiveViewExtensions.

Public Methods

	Name	Description
S	AsLive	Overloaded. Creates a view based on the specified IObservableSource<T> collection.
S	AsNonUpdatable<T>	Specifies a view as read-only.
S	AsUpdatable<T>	Specifies a view as updatable.
S	LiveAggregate	Overloaded. Applies an accumulator function over a view.
S	LiveAverage	Overloaded. Computes the average of a view of System.Int32 values.
S	LiveCount	Overloaded. A view representing the number of elements in a view.
S	LiveMax	Overloaded. Computes the maximum value of a view of System.Double values.
S	LiveMin	Overloaded. Computes the minimum value of a view of nullable System.Double values that are obtained by invoking a transform

		function on each element of the source view.
S	LiveSum	Overloaded. Computes the sum of a view of System.Int32 values.

Top

See Also

Reference

LiveViewExtensions Class
C1.LiveLinq Namespace

Methods
For a list of all members of this type, see LiveViewExtensions members.

Public Methods

	Name	Description
S	AsLive	Overloaded. Creates a view based on the specified IObservableSource<T> collection.
S	AsNonUpdatable<T>	Specifies a view as read-only.
S	AsUpdatable<T>	Specifies a view as updatable.
S	LiveAggregate	Overloaded. Applies an accumulator function over a view.
S	LiveAverage	Overloaded. Computes the average of a view of System.Int32 values.
S	LiveCount	Overloaded. A view representing the number of elements in a view.
S	LiveMax	Overloaded. Computes the maximum value of a view of System.Double values.
S	LiveMin	Overloaded. Computes the minimum value of a view of nullable System.Double values that are obtained by invoking a transform

		function on each element of the source view.
S	LiveSum	Overloaded. Computes the sum of a view of System.Int32 values.

Top

See Also

Reference

LiveViewExtensions Class
C1.LiveLinq Namespace

AsLive Method
Creates a view based on the specified IObservableSource<T> collection.

Overload List

Overload	Description
AsLive<T>(IObservableSource<T>)	Creates a view based on the specified IObservableSource<T> collection.
AsLive<T>(IObservableSource<T>,ViewOrder)	Creates a view based on the specified IObservableSource<T> collection.
AsLive<T>(IBindingList)	Creates a view based on the specified System.ComponentModel.IBindingList data source.
AsLive<T>(IBindingList,ViewOrder)	Creates a view based on the specified System.ComponentModel.IBindingList data source.
AsLive<T>(BindingList<T>)	A typed specialization of the AsLive<T>(IBindingList) method.
AsLive<T>(BindingList<T>,ViewOrder)	A typed specialization of the AsLive<T>(IBindingList,ViewOrder) method.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members

AsLive<T>(IObservableSource<T>) Method
The type of the elements in the collection.

The IObservableSource<T> collection to expose as a view.

Creates a view based on the specified IObservableSource<T> collection.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function AsLive(Of T)( _`
`  ByVal source As IObservableSource(Of T) _`
`) As View(Of T)`

C#

`public static View<T> AsLive<T>(`
`  IObservableSource<T> source`
`)`

Parameters

source

The IObservableSource<T> collection to expose as a view.

Type Parameters

T

The type of the elements in the collection.

Return Value

A view that contains the same elements as the IObservableSource<T>

Remarks

The resulting view may have its elements ordered differently than they are ordered in the *source* collection. Correspondingly, views built on this resulting view (for example, if you filter it with **Where**) will not preserve the source order either. If you need to preserve the source order, consider using the other **AsLive** overload where you can specify to what extent you need the order to be preserved.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

AsLive<T>(IObservableSource<T>,ViewOrder) Method
The type of the elements in the collection.

The IObservableSource<T> collection to expose as a view.

Specifies whether to preserve source item order.

Creates a view based on the specified IObservableSource<T> collection.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function AsLive(Of T)( _
   ByVal source As IObservableSource(Of T), _
   ByVal order As ViewOrder _
) As View(Of T)
```

C#

```
public static View<T> AsLive<T>(
```

```
    IObservableSource<T> source,
    ViewOrder order
)
```

Parameters

source

The IObservableSource<T> collection to expose as a view.

order

Specifies whether to preserve source item order.

Type Parameters

T

The type of the elements in the collection.

Return Value

A view that contains the same elements as the IObservableSource<T>

Remarks

If the *order* parameter specifies preserving item order, the order of items in the source is preserved, at a certain performance cost, in the resulting view and in views based on it (for example, if you filter it with **Where**).

Note that **Join** does not preserve source order. If you need to order a join result, use **OrderBy** after **Join**.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

The type of the elements in the view.

The System.ComponentModel.IBindingList data source to expose as a view.

Creates a view based on the specified System.ComponentModel.IBindingList data source.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function AsLive(Of T)( _
   ByVal source As IBindingList _
) As View(Of T)
```

C#

```
public static View<T> AsLive<T>(
   IBindingList source
)
```

Parameters

source

The System.ComponentModel.IBindingList data source to expose as a view.

Type Parameters

T

The type of the elements in the view.

Return Value

A view that contains the same elements as the System.ComponentModel.IBindingList.

Remarks

Use this method to build views from existing data sources. The only requirements for the data source is that it implements the standard data binding interface System.ComponentModel.IBindingList.

The resulting view may have its elements ordered differently than they are ordered in the *source* collection. Correspondingly, views built on this resulting view (for example, if you filter it with **Where**) will not preserve the source order either. If you need to preserve the

source order, consider using the other **AsLive** overload where you can specify to what extent you need the order to be preserved.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

AsLive<T>(IBindingList,ViewOrder) Method
The type of the elements in the view.

The System.ComponentModel.IBindingList data source to expose as a view.

Specifies whether to preserve source item order.

Creates a view based on the specified System.ComponentModel.IBindingList data source.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function AsLive(Of T)( _
   ByVal source As IBindingList, _
   ByVal order As ViewOrder _
) As View(Of T)
```

C#

```
public static View<T> AsLive<T>(
   IBindingList source,
   ViewOrder order
)
```

Parameters

source

The System.ComponentModel.IBindingList data source to expose as a view.

order

Specifies whether to preserve source item order.

Type Parameters

T

The type of the elements in the view.

Return Value

A view that contains the same elements as the System.ComponentModel.IBindingList.

Remarks

Use this method to build views from existing data sources. The only requirements for the data source is that it implements the standard data binding interface System.ComponentModel.IBindingList.

If the *order* parameter specifies preserving item order, the order of items in the source is preserved, at a certain performance cost, in the resulting view and in views based on it (for example, if you filter it with **Where**).

Note that **Join** does not preserve source order. If you need to order a join result, use **OrderBy** after **Join**.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

AsLive<T>(BindingList<T>) Method
The type of the elements in the view.

The BindingList data source to expose as a view.

A typed specialization of the AsLive<T>(IBindingList) method.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function AsLive(Of T)( _`
`  ByVal source As BindingList(Of T) _`
`) As View(Of T)`

C#

`public static View<T> AsLive<T>(`
`  BindingList<T> source`
`)`

Parameters

source

The BindingList data source to expose as a view.

Type Parameters

T

The type of the elements in the view.

Return Value

A view that contains the same elements as the BindingList.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

AsLive<T>(BindingList<T>,ViewOrder) Method
The type of the elements in the view.

The BindingList data source to expose as a view.

Specifies whether to preserve source item order.

A typed specialization of the AsLive<T>(IBindingList,ViewOrder) method.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function AsLive(Of T)( _
   ByVal source As BindingList(Of T), _
   ByVal order As ViewOrder _
) As View(Of T)
```

C#

```
public static View<T> AsLive<T>(
   BindingList<T> source,
   ViewOrder order
)
```

Parameters

source

The BindingList data source to expose as a view.

order

Specifies whether to preserve source item order.

Type Parameters

T

The type of the elements in the view.

Return Value

A view that contains the same elements as the BindingList.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

AsNonUpdatable<T> Method
The type of the elements in the view.

The view to specify as read-only.

Specifies a view as read-only.

Syntax

Visual Basic (Declaration)

```
Public Shared Function AsNonUpdatable(Of T)( _
   ByVal view As View(Of T) _
) As View(Of T)
```

C#

```
public static View<T> AsNonUpdatable<T>(
   View<T> view
)
```

Parameters

view

The view to specify as read-only.

Type Parameters

T

The type of the elements in the view.

Return Value

The same *view*.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
AsUpdatable<T> Method

AsUpdatable<T> Method
The type of the elements in the view.

The view to specify as updatable.

Specifies a view as updatable.

Syntax

Visual Basic (Declaration)

`Public Shared Function AsUpdatable(Of T)( _`
`  ByVal view As View(Of T) _`
`) As View(Of T)`

C#

`public static View<T> AsUpdatable<T>(`
`  View<T> view`
`)`

Parameters

view

The view to specify as updatable.

Type Parameters

T

The type of the elements in the view.

Return Value

The same *view*.

Remarks

This method is used with **Join** operation to specify which of the two parts of the join you want to be updatable.

Properties exposed by a view can be updatable or read-only. Updatable properties of a view can be modified directly in the view. Read-only properties of a view cannot be modified directly in the view, but they still reflect up-to-date values of the source, so the difference is often not critical, you can always modify corresponding property in the source, that will automatically change the property in the view.

Only one of the two arguments of a **Join** can be updatable, the one you qualified with **AsUpdatable()**. In absence of this qualification, both parts of the join are read-only. For example, in `from c in customers.AsLive() join o in orders.AsLive().AsUpdatable() on o.CustomerID equals c.CustomerID select new { c.CustomerName, o.OrderDate, o.OrderAmount }` **CustomerName** is read-only, and **OrderDate** and **OrderAmount** are updatable.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
IsReadOnly Property
ViewRow Class
ViewRowState Enumeration

LiveAggregate Method
Applies an accumulator function over a view.

Overload List

Overload	Description
LiveAggregate<TSource>(View<TSource>,Expression<Func<TSource,TSource,TSource>>,Expression<Func<TSource,TSource,TSource>>,Expression<Func<TSource,TSource,Boolean>>)	Applies an accumulator function over a view.
LiveAggregate<TSource,TAccumulate>(View<TSource>,TAccumulate,Expression<Func<TAccumulate,TSource,TAccumulate>>,Expression<Func<TAccumulate,TSource,TAccumulate>>,Expression<Func<TAccumulate,TSource,Boolean>>)	Applies an accumulator function over a view.

	The specified seed value is used as the initial accumulator value.
LiveAggregate<TSource,TAccumulate,TResult>(View<TSource>,TAccumulate,Expression<Func<TAccumulate,TSource,TAccumulate>>,Expression<Func<TAccumulate,TSource,TAccumulate>>,Expression<Func<TAccumulate,TSource,Boolean>>,Expression<Func<TAccumulate,TResult>>)	Applies an accumulator function over a view.

	The specified seed value is used as the initial accumulator value, and the specified function is used to select the result

	valu e.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members

LiveAggregate<TSource>(View<TSource>,Expression<Func<TSource,TSource,TSource>>,Expression<Func<TSource,TSource,TSource>>,Expression<Func<TSource,TSource,Boolean>>) Method
The type of the elements of *source*.

A view to aggregate over.

An accumulator function to be invoked on each element that is added to the source view.

A function to be applied to the accumulated value and to an element to obtain the changed accumulated value, when an element is removed from the source view.

A function used to determine whether *funcRemove* must be applied when an element is removed from the source view, or the accumulated value is not affected by its removal.

Applies an accumulator function over a view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveAggregate(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal funcAdd As Expression(Of Func(Of TSource,TSource,TSource)), _
   ByVal funcRemove As Expression(Of Func(Of TSource,TSource,TSource)), _
   ByVal funcRemoveDefined As Expression(Of Func(Of TSource,TSource,Boolean)) _
) As AggregationView(Of TSource,TSource)
```

C#

```
public static AggregationView<TSource,TSource> LiveAggregate<TSource>(
   View<TSource> source,
   Expression<Func<TSource,TSource,TSource>> funcAdd,
   Expression<Func<TSource,TSource,TSource>> funcRemove,
   Expression<Func<TSource,TSource,bool>> funcRemoveDefined
)
```

Parameters

source

A view to aggregate over.

funcAdd

An accumulator function to be invoked on each element that is added to the source view.

funcRemove

A function to be applied to the accumulated value and to an element to obtain the changed accumulated value, when an element is removed from the source view.

funcRemoveDefined

A function used to determine whether *funcRemove* must be applied when an element is removed from the source view, or the accumulated value is not affected by its removal.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the final accumulator value.

Remarks

It is possible to use standard LINQ query operator **Aggregate** instead of **LiveAggregate**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Aggregate** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAggregate** will use a more

performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAggregate<TSource,TAccumulate>(View<TSource>,TAccumulate,Expression<Func<TAccumulate,TSource,TAccumulate>>,Expression<Func<TAccumulate,TSource,TAccumulate>>,Expression<Func<TAccumulate,TSource,Boolean>>) Method

The type of the elements of *source*.

The type of the accumulator value.

A view to aggregate over.

The initial accumulator value.

An accumulator function to be invoked on each element that is added to the source view.

A function to be applied to the accumulated value and to an element to obtain the changed accumulated value, when an element is removed from the source view.

A function used to determine whether *funcRemove* must be applied when an element is removed from the source view, or the accumulated value is not affected by its removal.

Applies an accumulator function over a view. The specified seed value is used as the initial accumulator value.

Syntax

Visual Basic (Declaration)	
`Public Overloads Shared Function LiveAggregate`	

```
    (Of TSource,TAccumulate)( _
   ByVal source As View(Of TSource), _
   ByVal seed As TAccumulate, _
   ByVal funcAdd As Expression(Of Func(Of TAccumulate,TSource,TAccumulate)), _
   ByVal funcRemove As Expression(Of Func(Of TAccumulate,TSource,TAccumulate)),
_
   ByVal funcRemoveDefined As Expression(Of Func(Of
TAccumulate,TSource,Boolean)) _
) As AggregationView(Of TSource,TAccumulate)
```

C#

```
public static AggregationView<TSource,TAccumulate>
LiveAggregate<TSource,TAccumulate>(
   View<TSource> source,
   TAccumulate seed,
   Expression<Func<TAccumulate,TSource,TAccumulate>> funcAdd,
   Expression<Func<TAccumulate,TSource,TAccumulate>> funcRemove,
   Expression<Func<TAccumulate,TSource,bool>> funcRemoveDefined
)
```

Parameters

source

A view to aggregate over.

seed

The initial accumulator value.

funcAdd

An accumulator function to be invoked on each element that is added to the source view.

funcRemove

A function to be applied to the accumulated value and to an element to obtain the changed accumulated value, when an element is removed from the source view.

funcRemoveDefined

A function used to determine whether *funcRemove* must be applied when an element is removed from the source view, or the accumulated value is not affected by its removal.

Type Parameters

TSource

The type of the elements of *source*.

TAccumulate

The type of the accumulator value.

Return Value

A view representing the final accumulator value.

Remarks

It is possible to use standard LINQ query operator **Aggregate** instead of **LiveAggregate**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Aggregate** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAggregate** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAggregate<TSource,TAccumulate,TResult>(View<TSource>,TAccumulate,Expression<Func<TAccumulate,TSource,TAccumulate>>,Expression<Func<TAccumulate,TSource,TAccumulate>>,Expression<Func<TAccumulate,TSource,Boolean>>,Expression<Func<TAccumulate,TResult>>) Method

The type of the elements of *source*.

The type of the accumulator value.

The type of the resulting value.

A view to aggregate over.

The initial accumulator value.

An accumulator function to be invoked on each element that is added to the source view.

A function to be applied to the accumulated value and to an element to obtain the changed accumulated value, when an element is removed from the source view.

A function used to determine whether *funcRemove* must be applied when an element is removed from the source view, or the accumulated value is not affected by its removal.

A function to transform the final accumulator value into the result value.

Applies an accumulator function over a view. The specified seed value is used as the initial accumulator value, and the specified function is used to select the result value.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveAggregate
    (Of TSource,TAccumulate,TResult)( _
   ByVal source As View(Of TSource), _
   ByVal seed As TAccumulate, _
   ByVal funcAdd As Expression(Of Func(Of TAccumulate,TSource,TAccumulate)), _
   ByVal funcRemove As Expression(Of Func(Of TAccumulate,TSource,TAccumulate)),
_
   ByVal funcRemoveDefined As Expression(Of Func(Of
TAccumulate,TSource,Boolean)), _
   ByVal resultSelector As Expression(Of Func(Of TAccumulate,TResult)) _
) As AggregationView(Of TSource,TResult)
```

C#

```
public static AggregationView<TSource,TResult>
LiveAggregate<TSource,TAccumulate,TResult>(
   View<TSource> source,
   TAccumulate seed,
   Expression<Func<TAccumulate,TSource,TAccumulate>> funcAdd,
   Expression<Func<TAccumulate,TSource,TAccumulate>> funcRemove,
```

```
    Expression<Func<TAccumulate,TSource,bool>> funcRemoveDefined,
    Expression<Func<TAccumulate,TResult>> resultSelector
)
```

Parameters

source

A view to aggregate over.

seed

The initial accumulator value.

funcAdd

An accumulator function to be invoked on each element that is added to the source view.

funcRemove

A function to be applied to the accumulated value and to an element to obtain the changed accumulated value, when an element is removed from the source view.

funcRemoveDefined

A function used to determine whether *funcRemove* must be applied when an element is removed from the source view, or the accumulated value is not affected by its removal.

resultSelector

A function to transform the final accumulator value into the result value.

Type Parameters

TSource

The type of the elements of *source*.

TAccumulate

The type of the accumulator value.

TResult

The type of the resulting value.

Return Value

A view representing the final accumulator value.

Remarks

It is possible to use standard LINQ query operator **Aggregate** instead of **LiveAggregate**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Aggregate** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAggregate** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage Method
Computes the average of a view of System.Int32 values.

Overload List

Overload	Description
LiveAverage(View<Int32>)	Computes the average of a view of System.Int32 values.

LiveAverage(View<Nullable<Int32>>)	Computes the average of a view of nullable System.Int32 values.
LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Int32>>)	Computes the average of a view of System.Int32 values that are obtained by invoking a transform function on each element of the source view.
LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Int32>>>)	Computes the average of a view of nullable System.Int32 values that are obtained by invoking a transform function on each element of the source view.

LiveAverage(View<Int64>)	Computes the average of a view of System.Int64 values.
LiveAverage(View<Nullable<Int64>>)	Computes the average of a view of nullable System.Int64 values.
LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Int64>>)	Computes the average of a view of System.Int64 values that are obtained by invoking a transform function on each element of the source view.
LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Int64>>>)	Computes the average of a view of nullable System.Int64 values that are obtained by invoking a

	transform function on each element of the source view.
LiveAverage(View<Decimal>)	Computes the average of a view of System.Decimal values.
LiveAverage(View<Nullable<Decimal>>)	Computes the average of a view of nullable System.Decimal values.
LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Decimal>>)	Computes the average of a view of System.Decimal values that are obtained by invoking a transform function on each element of the source view.
LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Decimal>>>)	Computes the average of a

	view of nullable System.Decimal values that are obtained by invoking a transform function on each element of the source view.
LiveAverage(View<Double>)	Computes the average of a view of System.Double values.
LiveAverage(View<Nullable<Double>>)	Computes the average of a view of nullable System.Double values.
LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Double>>)	Computes the average of a view of System.Double values that are obtained by invoking a transform function on

	each element of the source view.
LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Double>>>)	Computes the average of a view of nullable System.Double values that are obtained by invoking a transform function on each element of the source view.
LiveAverage(View<Single>)	Computes the average of a view of System.Single values.
LiveAverage(View<Nullable<Single>>)	Computes the average of a view of nullable System.Single values.
LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Single>>)	Computes the average of a view of

	System.Single values that are obtained by invoking a transform function on each element of the source view.
LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Single>>>)	Computes the average of a view of nullable System.Single values that are obtained by invoking a transform function on each element of the source view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members

LiveAverage(View<Int32>) Method

A view containing the values to calculate the average of.

Computes the average of a view of System.Int32 values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveAverage( _
   ByVal source As View(Of Integer) _
) As AggregationView(Of Integer,Double)
```

C#

```
public static AggregationView<int,double> LiveAverage(
   View<int> source
)
```

Parameters

source

A view containing the values to calculate the average of.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty or contains only nulls, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage(View<Nullable<Int32>>) Method
A view containing the values to calculate the average of.

Computes the average of a view of nullable System.Int32 values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveAverage( _
   ByVal source As View(Of Nullable(Of Integer)) _
) As AggregationView(Of Nullable(Of Integer),Nullable(Of Double))
```

C#

```
public static AggregationView<Nullable<int>,Nullable<double>> LiveAverage(
   View<Nullable<int>> source
)
```

Parameters

source

A view containing the values to calculate the average of.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty or contains only nulls, the average value is null.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant

algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Int32>>) Method
The type of the elements of *source*.

A view containing the values to calculate the average of.

A transform function to apply to each element.

Computes the average of a view of System.Int32 values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveAverage(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Integer)) _
) As AggregationView(Of TSource,Double)
```

C#

```
public static AggregationView<TSource,double> LiveAverage<TSource>(
   View<TSource> source,
   Expression<Func<TSource,int>> selector
)
```

Parameters

source

A view containing the values to calculate the average of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Int32>>>) Method
The type of the elements of *source*.

A view containing the values to calculate the average of.

A transform function to apply to each element.

Computes the average of a view of nullable System.Int32 values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveAverage(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Integer))) _
) As AggregationView(Of TSource,Nullable(Of Double))
```

C#

```
public static AggregationView<TSource,Nullable<double>> LiveAverage<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<int>>> selector
)
```

Parameters

source

A view containing the values to calculate the average of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty, the average value is null.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage(View<Int64>) Method
A view containing the values to calculate the average of.

Computes the average of a view of System.Int64 values.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function LiveAverage( _`
`  ByVal source As View(Of Long) _`
`) As AggregationView(Of Long,Double)`

C#

`public static AggregationView<long,double> LiveAverage(`
`  View<long> source`
`)`

Parameters

source

A view containing the values to calculate the average of.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty or contains only nulls, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage(View<Nullable<Int64>>) Method
A view containing the values to calculate the average of.

Computes the average of a view of nullable System.Int64 values.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function LiveAverage( _`
`  ByVal source As View(Of Nullable(Of Long)) _`
`) As AggregationView(Of Nullable(Of Long),Nullable(Of Double))`

C#

```
public static AggregationView<Nullable<long>,Nullable<double>> LiveAverage(
   View<Nullable<long>> source
)
```

Parameters

source

A view containing the values to calculate the average of.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty or contains only nulls, the average value is null.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Int64>>) Method
The type of the elements of *source*.

A view containing the values to calculate the average of.

A transform function to apply to each element.

Computes the average of a view of System.Int64 values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveAverage(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Long)) _
) As AggregationView(Of TSource,Double)
```

C#

```
public static AggregationView<TSource,double> LiveAverage<TSource>(
   View<TSource> source,
   Expression<Func<TSource,long>> selector
)
```

Parameters

source

A view containing the values to calculate the average of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every

time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Int64>>>) Method
The type of the elements of *source*.

A view containing the values to calculate the average of.

A transform function to apply to each element.

Computes the average of a view of nullable System.Int64 values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveAverage(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Long))) _
) As AggregationView(Of TSource,Nullable(Of Double))
```

C#

```
public static AggregationView<TSource,Nullable<double>> LiveAverage<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<long>>> selector
)
```

Parameters

source

A view containing the values to calculate the average of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty, the average value is null.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage(View<Decimal>) Method
A view containing the values to calculate the average of.

Computes the average of a view of System.Decimal values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveAverage( _
   ByVal source As View(Of Decimal) _
) As AggregationView(Of Decimal,Decimal)
```

C#

```
public static AggregationView<decimal,decimal> LiveAverage(
   View<decimal> source
)
```

Parameters

source

A view containing the values to calculate the average of.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty or contains only nulls, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage(View<Nullable<Decimal>>) Method
A view containing the values to calculate the average of.

Computes the average of a view of nullable System.Decimal values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveAverage( _
   ByVal source As View(Of Nullable(Of Decimal)) _
) As AggregationView(Of Nullable(Of Decimal),Nullable(Of Decimal))
```

C#

```
public static AggregationView<Nullable<decimal>,Nullable<decimal>> LiveAverage(
   View<Nullable<decimal>> source
)
```

Parameters

source

A view containing the values to calculate the average of.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty or contains only nulls, the average value is null.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Decimal>>) Method
The type of the elements of *source*.

A view containing the values to calculate the average of.

A transform function to apply to each element.

Computes the average of a view of System.Decimal values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveAverage(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Decimal)) _
) As AggregationView(Of TSource,Decimal)
```

C#

```
public static AggregationView<TSource,decimal> LiveAverage<TSource>(
   View<TSource> source,
   Expression<Func<TSource,decimal>> selector
)
```

Parameters

source

A view containing the values to calculate the average of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Decimal>>>) Method
The type of the elements of *source*.

A view containing the values to calculate the average of.

A transform function to apply to each element.

Computes the average of a view of nullable System.Decimal values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveAverage(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Decimal))) _
) As AggregationView(Of TSource,Nullable(Of Decimal))
```

C#

```
public static AggregationView<TSource,Nullable<decimal>> LiveAverage<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<decimal>>> selector
)
```

Parameters

source

A view containing the values to calculate the average of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty, the average value is null.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage(View<Double>) Method
A view containing the values to calculate the average of.

Computes the average of a view of System.Double values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveAverage( _
   ByVal source As View(Of Double) _
) As AggregationView(Of Double,Double)
```

C#

```
public static AggregationView<double,double> LiveAverage(
   View<double> source
)
```

Parameters

source

A view containing the values to calculate the average of.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty or contains only nulls, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage(View<Nullable<Double>>) Method
A view containing the values to calculate the average of.

Computes the average of a view of nullable System.Double values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveAverage( _
   ByVal source As View(Of Nullable(Of Double)) _
) As AggregationView(Of Nullable(Of Double),Nullable(Of Double))
```

C#

```
public static AggregationView<Nullable<double>,Nullable<double>> LiveAverage(
   View<Nullable<double>> source
)
```

Parameters

source

A view containing the values to calculate the average of.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty or contains only nulls, the average value is null.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Double>>) Method
The type of the elements of *source*.

A view containing the values to calculate the average of.

A transform function to apply to each element.

Computes the average of a view of System.Double values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)	

```
Public Overloads Shared Function LiveAverage(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Double)) _
) As AggregationView(Of TSource,Double)
```

C#

```
public static AggregationView<TSource,double> LiveAverage<TSource>(
   View<TSource> source,
   Expression<Func<TSource,double>> selector
)
```

Parameters

source

A view containing the values to calculate the average of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Double>>>) Method
The type of the elements of *source*.

A view containing the values to calculate the average of.

A transform function to apply to each element.

Computes the average of a view of nullable System.Double values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveAverage(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Double))) _
) As AggregationView(Of TSource,Nullable(Of Double))
```

C#

```
public static AggregationView<TSource,Nullable<double>> LiveAverage<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<double>>> selector
)
```

Parameters

source

A view containing the values to calculate the average of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty, the average value is null.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage(View<Single>) Method
A view containing the values to calculate the average of.

Computes the average of a view of System.Single values.

Syntax

Visual Basic (Declaration)	

```
Public Overloads Shared Function LiveAverage( _
   ByVal source As View(Of Single) _
) As AggregationView(Of Single,Double)
```

C#

```
public static AggregationView<float,double> LiveAverage(
   View<float> source
)
```

Parameters

source

A view containing the values to calculate the average of.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty or contains only nulls, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage(View<Nullable<Single>>) Method

A view containing the values to calculate the average of.

Computes the average of a view of nullable System.Single values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveAverage( _
   ByVal source As View(Of Nullable(Of Single)) _
) As AggregationView(Of Nullable(Of Single),Nullable(Of Double))
```

C#

```
public static AggregationView<Nullable<float>,Nullable<double>> LiveAverage(
   View<Nullable<float>> source
)
```

Parameters

source

A view containing the values to calculate the average of.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty or contains only nulls, the average value is null.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Single>>) Method
The type of the elements of *source*.

A view containing the values to calculate the average of.

A transform function to apply to each element.

Computes the average of a view of System.Single values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveAverage(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Single)) _
) As AggregationView(Of TSource,Double)
```

C#

```
public static AggregationView<TSource,double> LiveAverage<TSource>(
   View<TSource> source,
   Expression<Func<TSource,float>> selector
)
```

Parameters

source

A view containing the values to calculate the average of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Single>>>) Method
The type of the elements of *source*.

A view containing the values to calculate the average of.

A transform function to apply to each element.

Computes the average of a view of nullable System.Single values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveAverage(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Single))) _
) As AggregationView(Of TSource,Nullable(Of Double))
```

C#

```
public static AggregationView<TSource,Nullable<double>> LiveAverage<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<float>>> selector
)
```

Parameters

source

A view containing the values to calculate the average of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty, the average value is null.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveCount Method
A view representing the number of elements in a view.

Overload List

Overload	Description
LiveCount<T>(View<T>)	A view representing the number of elements in a view.
LiveCount<T>(View<T>,Expression<Func<T,Boolean>>)	A view representing the number of elements of the specified view satisfying a condition.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members

LiveCount<T>(View<T>) Method
The type of the elements of *source*.

A view that contains elements to be counted.

A view representing the number of elements in a view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveCount(Of T)( _
   ByVal source As View(Of T) _
) As AggregationView(Of T,Integer)
```

C#

```
public static AggregationView<T,int> LiveCount<T>(
   View<T> source
)
```

Parameters

source

A view that contains elements to be counted.

Type Parameters

T

The type of the elements of *source*.

Remarks

It is possible to use standard LINQ query operator **Count** instead of **LiveCount**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Count** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveCount** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveCount<T>(View<T>,Expression<Func<T,Boolean>>) Method
The type of the elements of *source*.

A view that contains elements to be tested and counted.

A function to test each element for a condition.

A view representing the number of elements of the specified view satisfying a condition.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function LiveCount(Of T)( _`
`  ByVal source As View(Of T), _`
`  ByVal predicate As Expression(Of Func(Of T,Boolean)) _`
`) As AggregationView(Of T,Integer)`

C#

`public static AggregationView<T,int> LiveCount<T>(`
`  View<T> source,`
`  Expression<Func<T,bool>> predicate`
`)`

Parameters

source

A view that contains elements to be tested and counted.

predicate

A function to test each element for a condition.

Type Parameters

T

The type of the elements of *source*.

Remarks

It is possible to use standard LINQ query operator **Count** instead of **LiveCount**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Count** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveCount** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax Method
Computes the maximum value of a view of System.Double values.

Overload List

Overload	Description
LiveMax(View<Double>)	Computes the maximum value of a view of System.Double values.
LiveMax(View<Nullable<Double>>)	Computes the maximum value of a view

	of nullable System.Double values.
LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Double>>)	Computes the maximum value of a view of System.Double values that are obtained by invoking a transform function on each element of the source view.
LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Double>>>)	Computes the maximum value of a view of nullable System.Double values that are obtained by invoking a transform function on each element of the source view.
LiveMax(View<Single>)	Computes the maximum

	value of a view of System.Single values.
LiveMax(View<Nullable<Single>>)	Computes the maximum value of a view of nullable System.Single values.
LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Single>>)	Computes the maximum value of a view of System.Single values that are obtained by invoking a transform function on each element of the source view.
LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Single>>>)	Computes the maximum value of a view of nullable System.Single values that are obtained by invoking a

	transform function on each element of the source view.
LiveMax<TSource,TResult>(View<TSource>,Expression<Func<TSource,TResult>>)	Invokes a transform function on each element of a view of elements of a generic type and computes the maximum resulting value.
LiveMax<TSource>(View<TSource>)	Computes the maximum value of a view of elements of a generic type.
LiveMax(View<Int32>)	Computes the maximum value of a view of System.Int32 values.
LiveMax(View<Nullable<Int32>>)	Computes the maximum

	value of a view of nullable System.Int32 values.
LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Int32>>)	Computes the maximum value of a view of System.Int32 values that are obtained by invoking a transform function on each element of the source view.
LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Int32>>>)	Computes the maximum value of a view of nullable System.Int32 values that are obtained by invoking a transform function on each element of the source view.
LiveMax(View<Decimal>)	Computes the

	maximum value of a view of System.Decimal values.
LiveMax(View<Nullable<Decimal>>)	Computes the maximum value of a view of nullable System.Decimal values.
LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Decimal>>)	Computes the maximum value of a view of System.Decimal values that are obtained by invoking a transform function on each element of the source view.
LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Decimal>>>)	Computes the maximum value of a view of nullable System.Decimal values that are obtained

	by invoking a transform function on each element of the source view.
LiveMax(View<Int64>)	Computes the maximum value of a view of System.Int64 values.
LiveMax(View<Nullable<Int64>>)	Computes the maximum value of a view of nullable System.Int64 values.
LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Int64>>)	Computes the maximum value of a view of System.Int64 values that are obtained by invoking a transform function on each element of the source view.

LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Int64>>>)	Computes the maximum value of a view of nullable System.Int64 values that are obtained by invoking a transform function on each element of the source view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members

LiveMax(View<Double>) Method
A view containing the values to determine the maximum value of.

Computes the maximum value of a view of System.Double values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMax( _
   ByVal source As View(Of Double) _
) As AggregationView(Of Double,Double)
```

C#

```
public static AggregationView<double,double> LiveMax(
   View<double> source
)
```

Parameters

source

A view containing the values to determine the maximum value of.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty or contains only nulls, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax(View<Nullable<Double>>) Method
A view containing the values to determine the maximum value of.

Computes the maximum value of a view of nullable System.Double values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMax( _
   ByVal source As View(Of Nullable(Of Double)) _
) As AggregationView(Of Nullable(Of Double),Nullable(Of Double))
```

C#

```
public static AggregationView<Nullable<double>,Nullable<double>> LiveMax(
   View<Nullable<double>> source
)
```

Parameters

source

A view containing the values to determine the maximum value of.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty or contains only nulls, the maximum value is null.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Double>>) Method
The type of the elements of *source*.

A view containing the values to determine the maximum value of.

A transform function to apply to each element.

Computes the maximum value of a view of System.Double values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMax(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Double)) _
) As AggregationView(Of TSource,Double)
```

C#

```
public static AggregationView<TSource,double> LiveMax<TSource>(
   View<TSource> source,
   Expression<Func<TSource,double>> selector
)
```

Parameters

source

A view containing the values to determine the maximum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Double>>>) Method
The type of the elements of *source*.

A view containing the values to determine the maximum value of.

A transform function to apply to each element.

Computes the maximum value of a view of nullable System.Double values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMax(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Double))) _
```

```
) As AggregationView(Of TSource,Nullable(Of Double))
```

C#

```
public static AggregationView<TSource,Nullable<double>> LiveMax<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<double>>> selector
)
```

Parameters

source

A view containing the values to determine the maximum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty or contains only nulls, the maximum value is null.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax(View<Single>) Method
A view containing the values to determine the maximum value of.

Computes the maximum value of a view of System.Single values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMax( _
   ByVal source As View(Of Single) _
) As AggregationView(Of Single,Single)
```

C#

```
public static AggregationView<float,float> LiveMax(
   View<float> source
)
```

Parameters

source

A view containing the values to determine the maximum value of.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty or contains only nulls, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate

it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax(View<Nullable<Single>>) Method
A view containing the values to determine the maximum value of.

Computes the maximum value of a view of nullable System.Single values.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function LiveMax( _`
`  ByVal source As View(Of Nullable(Of Single)) _`
`) As AggregationView(Of Nullable(Of Single),Nullable(Of Single))`

C#

`public static AggregationView<Nullable<float>,Nullable<float>> LiveMax(`
`  View<Nullable<float>> source`
`)`

Parameters

source

A view containing the values to determine the maximum value of.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty or contains only nulls, the maximum value is null.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Single>>) Method
The type of the elements of *source*.

A view containing the values to determine the maximum value of.

A transform function to apply to each element.

Computes the maximum value of a view of System.Single values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMax(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Single)) _
) As AggregationView(Of TSource,Single)
```

C#

```
public static AggregationView<TSource,float> LiveMax<TSource>(
   View<TSource> source,
   Expression<Func<TSource,float>> selector
)
```

Parameters

source

A view containing the values to determine the maximum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Single>>>) Method
The type of the elements of *source*.

A view containing the values to determine the maximum value of.

A transform function to apply to each element.

Computes the maximum value of a view of nullable System.Single values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMax(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Single))) _
) As AggregationView(Of TSource,Nullable(Of Single))
```

C#

```
public static AggregationView<TSource,Nullable<float>> LiveMax<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<float>>> selector
)
```

Parameters

source

A view containing the values to determine the maximum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty or contains only nulls, the maximum value is null.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax<TSource,TResult>(View<TSource>,Expression<Func<TSource,TResult>>) Method
The type of the elements of *source*.

The type of the value returned by *selector*.

A view containing the values to determine the maximum value of.

A transform function to apply to each element.

Invokes a transform function on each element of a view of elements of a generic type and computes the maximum resulting value.

Syntax

Visual Basic (Declaration)	

```
Public Overloads Shared Function LiveMax
    (Of TSource,TResult)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,TResult)) _
) As AggregationView(Of TSource,TResult)
```

C#

```
public static AggregationView<TSource,TResult> LiveMax<TSource,TResult>(
   View<TSource> source,
   Expression<Func<TSource,TResult>> selector
)
```

Parameters

source

A view containing the values to determine the maximum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

TResult

The type of the value returned by *selector*.

Return Value

A view representing the maximum of the values.

Remarks

If type *TResult* implements IComparable, this method uses that implementation to compare values. Otherwise, if type *TResult* implements System.IComparable, that implementation is used to compare values.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas

LiveMax will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax<TSource>(View<TSource>) Method
The type of the elements of *source*.

A view containing the values to determine the maximum value of.

Computes the maximum value of a view of elements of a generic type.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMax(Of TSource)( _
   ByVal source As View(Of TSource) _
) As AggregationView(Of TSource,TSource)
```

C#

```
public static AggregationView<TSource,TSource> LiveMax<TSource>(
   View<TSource> source
)
```

Parameters

source

A view containing the values to determine the maximum value of.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the maximum of the values.

Remarks

If type *TSource* implements IComparable, this method uses that implementation to compare values. Otherwise, if type *TSource* implements System.IComparable, that implementation is used to compare values.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax(View<Int32>) Method
A view containing the values to determine the maximum value of.

Computes the maximum value of a view of System.Int32 values.

Syntax

Visual Basic (Declaration)	

```
Public Overloads Shared Function LiveMax( _
   ByVal source As View(Of Integer) _
) As AggregationView(Of Integer,Integer)
```

C#

```
public static AggregationView<int,int> LiveMax(
   View<int> source
)
```

Parameters

source

A view containing the values to determine the maximum value of.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty or contains only nulls, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax(View<Nullable<Int32>>) Method
A view containing the values to determine the maximum value of.

Computes the maximum value of a view of nullable System.Int32 values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMax( _
   ByVal source As View(Of Nullable(Of Integer)) _
) As AggregationView(Of Nullable(Of Integer),Nullable(Of Integer))
```

C#

```
public static AggregationView<Nullable<int>,Nullable<int>> LiveMax(
   View<Nullable<int>> source
)
```

Parameters

source

A view containing the values to determine the maximum value of.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty or contains only nulls, the maximum value is null.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Int32>>) Method
The type of the elements of *source*.

A view containing the values to determine the maximum value of.

A transform function to apply to each element.

Computes the maximum value of a view of System.Int32 values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMax(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Integer)) _
) As AggregationView(Of TSource,Integer)
```

C#

```
public static AggregationView<TSource,int> LiveMax<TSource>(
   View<TSource> source,
   Expression<Func<TSource,int>> selector
)
```

Parameters

source

A view containing the values to determine the maximum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Int32>>>) Method
The type of the elements of *source*.

A view containing the values to determine the maximum value of.

A transform function to apply to each element.

Computes the maximum value of a view of nullable System.Int32 values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMax(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Integer))) _
```

```
) As AggregationView(Of TSource,Nullable(Of Integer))
```

C#

```
public static AggregationView<TSource,Nullable<int>> LiveMax<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<int>>> selector
)
```

Parameters

source

A view containing the values to determine the maximum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty or contains only nulls, the maximum value is null.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax(View<Decimal>) Method
A view containing the values to determine the maximum value of.

Computes the maximum value of a view of System.Decimal values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMax( _
   ByVal source As View(Of Decimal) _
) As AggregationView(Of Decimal,Decimal)
```

C#

```
public static AggregationView<decimal,decimal> LiveMax(
   View<decimal> source
)
```

Parameters

source

A view containing the values to determine the maximum value of.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty or contains only nulls, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate

it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax(View<Nullable<Decimal>>) Method
A view containing the values to determine the maximum value of.

Computes the maximum value of a view of nullable System.Decimal values.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function LiveMax( _`
`  ByVal source As View(Of Nullable(Of Decimal)) _`
`) As AggregationView(Of Nullable(Of Decimal),Nullable(Of Decimal))`

C#

`public static AggregationView<Nullable<decimal>,Nullable<decimal>> LiveMax(`
`  View<Nullable<decimal>> source`
`)`

Parameters

source

A view containing the values to determine the maximum value of.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty or contains only nulls, the maximum value is null.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Decimal>>) Method
The type of the elements of *source*.

A view containing the values to determine the maximum value of.

A transform function to apply to each element.

Computes the maximum value of a view of System.Decimal values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMax(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Decimal)) _
) As AggregationView(Of TSource,Decimal)
```

C#

```
public static AggregationView<TSource,decimal> LiveMax<TSource>(
   View<TSource> source,
   Expression<Func<TSource,decimal>> selector
)
```

Parameters

source

A view containing the values to determine the maximum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Decimal>>>) Method
The type of the elements of *source*.

A view containing the values to determine the maximum value of.

A transform function to apply to each element.

Computes the maximum value of a view of nullable System.Decimal values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMax(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Decimal))) _
) As AggregationView(Of TSource,Nullable(Of Decimal))
```

C#

```
public static AggregationView<TSource,Nullable<decimal>> LiveMax<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<decimal>>> selector
)
```

Parameters

source

A view containing the values to determine the maximum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty or contains only nulls, the maximum value is null.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax(View<Int64>) Method
A view containing the values to determine the maximum value of.

Computes the maximum value of a view of System.Int64 values.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function LiveMax( _`
`  ByVal source As View(Of Long) _`
`) As AggregationView(Of Long,Long)`

C#

```
public static AggregationView<long,long> LiveMax(
   View<long> source
)
```

Parameters

source

A view containing the values to determine the maximum value of.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty or contains only nulls, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax(View<Nullable<Int64>>) Method
A view containing the values to determine the maximum value of.

Computes the maximum value of a view of nullable System.Int64 values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMax( _
   ByVal source As View(Of Nullable(Of Long)) _
) As AggregationView(Of Nullable(Of Long),Nullable(Of Long))
```

C#

```
public static AggregationView<Nullable<long>,Nullable<long>> LiveMax(
   View<Nullable<long>> source
)
```

Parameters

source

A view containing the values to determine the maximum value of.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty or contains only nulls, the maximum value is null.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

The type of the elements of *source*.

A view containing the values to determine the maximum value of.

A transform function to apply to each element.

Computes the maximum value of a view of System.Int64 values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMax(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Long)) _
) As AggregationView(Of TSource,Long)
```

C#

```
public static AggregationView<TSource,long> LiveMax<TSource>(
   View<TSource> source,
   Expression<Func<TSource,long>> selector
)
```

Parameters

source

A view containing the values to determine the maximum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Int64>>>) Method
The type of the elements of *source*.

A view containing the values to determine the maximum value of.

A transform function to apply to each element.

Computes the maximum value of a view of nullable System.Int64 values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMax(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Long))) _
) As AggregationView(Of TSource,Nullable(Of Long))
```

C#

```
public static AggregationView<TSource,Nullable<long>> LiveMax<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<long>>> selector
)
```

Parameters

source

A view containing the values to determine the maximum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty or contains only nulls, the maximum value is null.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin Method

Computes the minimum value of a view of nullable System.Double values that are obtained by invoking a transform function on each element of the source view.

Overload List

Overload	Description
LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Double>>>)	Computes the minimum value of a view of nullable System.Double values that are obtained by invoking a transform function on each element of the source view.
LiveMin(View<Single>)	Computes the minimum value of a view of System.Single values.
LiveMin(View<Nullable<Single>>)	Computes the minimum value of a view of nullable

	System.Single values.
LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Single>>)	Computes the minimum value of a view of System.Single values that are obtained by invoking a transform function on each element of the source view.
LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Single>>>)	Computes the minimum value of a view of nullable System.Single values that are obtained by invoking a transform function on each element of the source view.
LiveMin<TSource,TResult>(View<TSource>,Expression<Func<TSource,TResult>>)	Invokes a transform function on

	each element of a view of elements of a generic type and computes the minimum resulting value.
LiveMin<TSource>(View<TSource>)	Computes the minimum value of a view of elements of a generic type.
LiveMin(View<Int32>)	Computes the minimum value of a view of System.Int32 values.
LiveMin(View<Nullable<Int32>>)	Computes the minimum value of a view of nullable System.Int32 values.
LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Int32>>)	Computes the minimum value of a view of

	System.Int32 values that are obtained by invoking a transform function on each element of the source view.
LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Int32>>>)	Computes the minimum value of a view of nullable System.Int32 values that are obtained by invoking a transform function on each element of the source view.
LiveMin(View<Decimal>)	Computes the minimum value of a view of System.Decimal values.
LiveMin(View<Nullable<Decimal>>)	Computes the minimum value of a view

	of nullable System.Decimal values.
LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Decimal>>)	Computes the minimum value of a view of System.Decimal values that are obtained by invoking a transform function on each element of the source view.
LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Decimal>>>)	Computes the minimum value of a view of nullable System.Decimal values that are obtained by invoking a transform function on each element of the source view.
LiveMin(View<Int64>)	Computes the minimum

	value of a view of System.Int64 values.
LiveMin(View<Nullable<Int64>>)	Computes the minimum value of a view of nullable System.Int64 values.
LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Int64>>)	Computes the minimum value of a view of System.Int64 values that are obtained by invoking a transform function on each element of the source view.
LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Int64>>>)	Computes the minimum value of a view of nullable System.Int64 values that are obtained by invoking a

	transform function on each element of the source view.
LiveMin(View<Double>)	Computes the minimum value of a view of System.Double values.
LiveMin(View<Nullable<Double>>)	Computes the minimum value of a view of nullable System.Double values.
LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Double>>)	Computes the minimum value of a view of System.Double values that are obtained by invoking a transform function on each element of the source view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members

LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Double>>>) Method
The type of the elements of *source*.

A view containing the values to determine the minimum value of.

A transform function to apply to each element.

Computes the minimum value of a view of nullable System.Double values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMin(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Double))) _
) As AggregationView(Of TSource,Nullable(Of Double))
```

C#

```
public static AggregationView<TSource,Nullable<double>> LiveMin<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<double>>> selector
)
```

Parameters

source

A view containing the values to determine the minimum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty or contains only nulls, the minimum value is null.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin(View<Single>) Method
A view containing the values to determine the minimum value of.

Computes the minimum value of a view of System.Single values.

Syntax

Visual Basic (Declaration)	
`Public Overloads Shared Function LiveMin( _`	

```
   ByVal source As View(Of Single) _
) As AggregationView(Of Single,Single)
```

C#

```
public static AggregationView<float,float> LiveMin(
   View<float> source
)
```

Parameters

source

A view containing the values to determine the minimum value of.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty or contains only nulls, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin(View<Nullable<Single>>) Method
A view containing the values to determine the minimum value of.

Computes the minimum value of a view of nullable System.Single values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMin( _
   ByVal source As View(Of Nullable(Of Single)) _
) As AggregationView(Of Nullable(Of Single),Nullable(Of Single))
```

C#

```
public static AggregationView<Nullable<float>,Nullable<float>> LiveMin(
   View<Nullable<float>> source
)
```

Parameters

source

A view containing the values to determine the minimum value of.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty or contains only nulls, the minimum value is null.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Single>>) Method
The type of the elements of *source*.

A view containing the values to determine the minimum value of.

A transform function to apply to each element.

Computes the minimum value of a view of System.Single values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function LiveMin(Of TSource)( _`
`  ByVal source As View(Of TSource), _`
`  ByVal selector As Expression(Of Func(Of TSource,Single)) _`
`) As AggregationView(Of TSource,Single)`

C#

`public static AggregationView<TSource,float> LiveMin<TSource>(`
`  View<TSource> source,`
`  Expression<Func<TSource,float>> selector`
`)`

Parameters

source

A view containing the values to determine the minimum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Single>>>) Method
The type of the elements of *source*.

A view containing the values to determine the minimum value of.

A transform function to apply to each element.

Computes the minimum value of a view of nullable System.Single values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMin(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Single))) _
```

```
) As AggregationView(Of TSource,Nullable(Of Single))
```

C#

```
public static AggregationView<TSource,Nullable<float>> LiveMin<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<float>>> selector
)
```

Parameters

source

A view containing the values to determine the minimum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty or contains only nulls, the minimum value is null.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin<TSource,TResult>(View<TSource>,Expression<Func<TSource,TResult>>) Method
The type of the elements of *source*.

The type of the value returned by *selector*.

A view containing the values to determine the minimum value of.

A transform function to apply to each element.

Invokes a transform function on each element of a view of elements of a generic type and computes the minimum resulting value.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMin
    (Of TSource,TResult)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,TResult)) _
) As AggregationView(Of TSource,TResult)
```

C#

```
public static AggregationView<TSource,TResult> LiveMin<TSource,TResult>(
   View<TSource> source,
   Expression<Func<TSource,TResult>> selector
)
```

Parameters

source

A view containing the values to determine the minimum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

TResult

The type of the value returned by *selector*.

Return Value

A view representing the minimum of the values.

Remarks

If type *TResult* implements IComparable, this method uses that implementation to compare values. Otherwise, if type *TResult* implements System.IComparable, that implementation is used to compare values.

It is possible to use standard LINQ query operator **Min** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin<TSource>(View<TSource>) Method
The type of the elements of *source*.

A view containing the values to determine the minimum value of.

Computes the minimum value of a view of elements of a generic type.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMin(Of TSource)( _
   ByVal source As View(Of TSource) _
) As AggregationView(Of TSource,TSource)
```

C#

```
public static AggregationView<TSource,TSource> LiveMin<TSource>(
   View<TSource> source
)
```

Parameters

source

A view containing the values to determine the minimum value of.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the minimum of the values.

Remarks

If type *TSource* implements IComparable, this method uses that implementation to compare values. Otherwise, if type *TSource* implements System.IComparable, that implementation is used to compare values.

It is possible to use standard LINQ query operator **Min** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin(View<Int32>) Method
A view containing the values to determine the minimum value of.

Computes the minimum value of a view of System.Int32 values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMin( _
   ByVal source As View(Of Integer) _
) As AggregationView(Of Integer,Integer)
```

C#

```
public static AggregationView<int,int> LiveMin(
   View<int> source
)
```

Parameters

source

A view containing the values to determine the minimum value of.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty or contains only nulls, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin(View<Nullable<Int32>>) Method
A view containing the values to determine the minimum value of.

Computes the minimum value of a view of nullable System.Int32 values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMin( _
   ByVal source As View(Of Nullable(Of Integer)) _
) As AggregationView(Of Nullable(Of Integer),Nullable(Of Integer))
```

C#

```
public static AggregationView<Nullable<int>,Nullable<int>> LiveMin(
   View<Nullable<int>> source
)
```

Parameters

source

A view containing the values to determine the minimum value of.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty or contains only nulls, the minimum value is null.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Int32>>) Method
The type of the elements of *source*.

A view containing the values to determine the minimum value of.

A transform function to apply to each element.

Computes the minimum value of a view of System.Int32 values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)
`Public Overloads Shared Function LiveMin(Of TSource)( _`

```
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Integer)) _
) As AggregationView(Of TSource,Integer)
```

C#

```
public static AggregationView<TSource,int> LiveMin<TSource>(
   View<TSource> source,
   Expression<Func<TSource,int>> selector
)
```

Parameters

source

A view containing the values to determine the minimum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Int32>>>) Method
The type of the elements of *source*.

A view containing the values to determine the minimum value of.

A transform function to apply to each element.

Computes the minimum value of a view of nullable System.Int32 values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMin(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Integer))) _
) As AggregationView(Of TSource,Nullable(Of Integer))
```

C#

```
public static AggregationView<TSource,Nullable<int>> LiveMin<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<int>>> selector
)
```

Parameters

source

A view containing the values to determine the minimum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty or contains only nulls, the minimum value is null.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin(View<Decimal>) Method
A view containing the values to determine the minimum value of.

Computes the minimum value of a view of System.Decimal values.

Syntax

Visual Basic (Declaration)	

```
Public Overloads Shared Function LiveMin( _
   ByVal source As View(Of Decimal) _
) As AggregationView(Of Decimal,Decimal)
```

C#

```
public static AggregationView<decimal,decimal> LiveMin(
   View<decimal> source
)
```

Parameters

source

A view containing the values to determine the minimum value of.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty or contains only nulls, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin(View<Nullable<Decimal>>) Method
A view containing the values to determine the minimum value of.

Computes the minimum value of a view of nullable System.Decimal values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMin( _
   ByVal source As View(Of Nullable(Of Decimal)) _
) As AggregationView(Of Nullable(Of Decimal),Nullable(Of Decimal))
```

C#

```
public static AggregationView<Nullable<decimal>,Nullable<decimal>> LiveMin(
   View<Nullable<decimal>> source
)
```

Parameters

source

A view containing the values to determine the minimum value of.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty or contains only nulls, the minimum value is null.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Decimal>>) Method
The type of the elements of *source*.

A view containing the values to determine the minimum value of.

A transform function to apply to each element.

Computes the minimum value of a view of System.Decimal values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMin(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Decimal)) _
) As AggregationView(Of TSource,Decimal)
```

C#

```
public static AggregationView<TSource,decimal> LiveMin<TSource>(
   View<TSource> source,
   Expression<Func<TSource,decimal>> selector
)
```

Parameters

source

A view containing the values to determine the minimum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Decimal>>>) Method
The type of the elements of *source*.

A view containing the values to determine the minimum value of.

A transform function to apply to each element.

Computes the minimum value of a view of nullable System.Decimal values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMin(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Decimal))) _
```

```
) As AggregationView(Of TSource,Nullable(Of Decimal))
```

C#

```
public static AggregationView<TSource,Nullable<decimal>> LiveMin<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<decimal>>> selector
)
```

Parameters

source

A view containing the values to determine the minimum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty or contains only nulls, the minimum value is null.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin(View<Int64>) Method
A view containing the values to determine the minimum value of.

Computes the minimum value of a view of System.Int64 values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMin( _
   ByVal source As View(Of Long) _
) As AggregationView(Of Long,Long)
```

C#

```
public static AggregationView<long,long> LiveMin(
   View<long> source
)
```

Parameters

source

A view containing the values to determine the minimum value of.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty or contains only nulls, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate

it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin(View<Nullable<Int64>>) Method
A view containing the values to determine the minimum value of.

Computes the minimum value of a view of nullable System.Int64 values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMin( _
   ByVal source As View(Of Nullable(Of Long)) _
) As AggregationView(Of Nullable(Of Long),Nullable(Of Long))
```

C#

```
public static AggregationView<Nullable<long>,Nullable<long>> LiveMin(
   View<Nullable<long>> source
)
```

Parameters

source

A view containing the values to determine the minimum value of.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty or contains only nulls, the minimum value is null.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Int64>>) Method
The type of the elements of *source*.

A view containing the values to determine the minimum value of.

A transform function to apply to each element.

Computes the minimum value of a view of System.Int64 values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMin(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Long)) _
) As AggregationView(Of TSource,Long)
```

C#

```
public static AggregationView<TSource,long> LiveMin<TSource>(
   View<TSource> source,
   Expression<Func<TSource,long>> selector
)
```

Parameters

source

A view containing the values to determine the minimum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Int64>>>) Method
The type of the elements of *source*.

A view containing the values to determine the minimum value of.

A transform function to apply to each element.

Computes the minimum value of a view of nullable System.Int64 values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMin(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Long))) _
) As AggregationView(Of TSource,Nullable(Of Long))
```

C#

```
public static AggregationView<TSource,Nullable<long>> LiveMin<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<long>>> selector
)
```

Parameters

source

A view containing the values to determine the minimum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty or contains only nulls, the minimum value is null.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin(View<Double>) Method
A view containing the values to determine the minimum value of.

Computes the minimum value of a view of System.Double values.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function LiveMin( _`
`  ByVal source As View(Of Double) _`
`) As AggregationView(Of Double,Double)`

C#

```
public static AggregationView<double,double> LiveMin(
   View<double> source
)
```

Parameters

source

A view containing the values to determine the minimum value of.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty or contains only nulls, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin(View<Nullable<Double>>) Method
A view containing the values to determine the minimum value of.

Computes the minimum value of a view of nullable System.Double values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMin( _
   ByVal source As View(Of Nullable(Of Double)) _
) As AggregationView(Of Nullable(Of Double),Nullable(Of Double))
```

C#

```
public static AggregationView<Nullable<double>,Nullable<double>> LiveMin(
   View<Nullable<double>> source
)
```

Parameters

source

A view containing the values to determine the minimum value of.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty or contains only nulls, the minimum value is null.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

The type of the elements of *source*.

A view containing the values to determine the minimum value of.

A transform function to apply to each element.

Computes the minimum value of a view of System.Double values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function LiveMin(Of TSource)( _`
`  ByVal source As View(Of TSource), _`
`  ByVal selector As Expression(Of Func(Of TSource,Double)) _`
`) As AggregationView(Of TSource,Double)`

C#

`public static AggregationView<TSource,double> LiveMin<TSource>(`
`  View<TSource> source,`
`  Expression<Func<TSource,double>> selector`
`)`

Parameters

source

A view containing the values to determine the minimum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveSum Method
Computes the sum of a view of System.Int32 values.

Overload List

Overload	Description
LiveSum(View<Int32>)	Computes the sum of a view of System.Int32 values.
LiveSum(View<Nullable<Int32>>)	Computes the sum of a view of nullable System.Int32

	values.
LiveSum<TSource>(View<TSource>,Expression<Func<TSource,Int32>>)	Computes the sum of a view of System.Int32 values that are obtained by invoking a transform function on each element of the source view.
LiveSum<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Int32>>>)	Computes the sum of a view of nullable System.Int32 values that are obtained by invoking a transform function on each element of the source view.
LiveSum(View<Decimal>)	Computes the sum of a view of System.Decimal values.

LiveSum(View<Nullable<Decimal>>)	Computes the sum of a view of nullable System.Decimal values.
LiveSum<TSource>(View<TSource>,Expression<Func<TSource,Decimal>>)	Computes the sum of a view of System.Decimal values that are obtained by invoking a transform function on each element of the source view.
LiveSum<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Decimal>>>)	Computes the sum of a view of nullable System.Decimal values that are obtained by invoking a transform function on each element of the source view.
LiveSum(View<Int64>)	Computes the sum of a view

	of System.Int64 values.
LiveSum(View<Nullable<Int64>>)	Computes the sum of a view of nullable System.Int64 values.
LiveSum<TSource>(View<TSource>,Expression<Func<TSource,Int64>>)	Computes the sum of a view of System.Int64 values that are obtained by invoking a transform function on each element of the source view.
LiveSum<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Int64>>>)	Computes the sum of a view of nullable System.Int64 values that are obtained by invoking a transform function on each element of the source

	view.
LiveSum(View<Double>)	Computes the sum of a view of System.Double values.
LiveSum(View<Nullable<Double>>)	Computes the sum of a view of nullable System.Double values.
LiveSum<TSource>(View<TSource>,Expression<Func<TSource,Double>>)	Computes the sum of a view of System.Double values that are obtained by invoking a transform function on each element of the source view.
LiveSum<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Double>>>)	Computes the sum of a view of nullable System.Double values that are obtained by invoking a

	transform function on each element of the source view.
LiveSum(View<Single>)	Computes the sum of a view of System.Single values.
LiveSum(View<Nullable<Single>>)	Computes the sum of a view of nullable System.Single values.
LiveSum<TSource>(View<TSource>,Expression<Func<TSource,Single>>)	Computes the sum of a view of System.Single values that are obtained by invoking a transform function on each element of the source view.
LiveSum<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Single>>>)	Computes the sum of a view of nullable

	System.Single values that are obtained by invoking a transform function on each element of the source view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members

LiveSum(View<Int32>) Method
A view containing the values to calculate the sum of.

Computes the sum of a view of System.Int32 values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveSum( _
   ByVal source As View(Of Integer) _
) As AggregationView(Of Integer,Integer)
```

C#

```
public static AggregationView<int,int> LiveSum(
   View<int> source
)
```

Parameters

source

A view containing the values to calculate the sum of.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveSum(View<Nullable<Int32>>) Method
A view containing the values to calculate the sum of.

Computes the sum of a view of nullable System.Int32 values.

Syntax

Visual Basic (Declaration)	

```
Public Overloads Shared Function LiveSum( _
   ByVal source As View(Of Nullable(Of Integer)) _
```

```
) As AggregationView(Of Nullable(Of Integer),Nullable(Of Integer))
```

C#

```
public static AggregationView<Nullable<int>,Nullable<int>> LiveSum(
   View<Nullable<int>> source
)
```

Parameters

source

A view containing the values to calculate the sum of.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveSum<TSource>(View<TSource>,Expression<Func<TSource,Int32>>) Method
The type of the elements of *source*.

A view containing the values to calculate the sum of.

A transform function to apply to each element.

Computes the sum of a view of System.Int32 values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveSum(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Integer)) _
) As AggregationView(Of TSource,Integer)
```

C#

```
public static AggregationView<TSource,int> LiveSum<TSource>(
   View<TSource> source,
   Expression<Func<TSource,int>> selector
)
```

Parameters

source

A view containing the values to calculate the sum of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveSum<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Int32>>>) Method
The type of the elements of *source*.

A view containing the values to calculate the sum of.

A transform function to apply to each element.

Computes the sum of a view of nullable System.Int32 values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveSum(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Integer))) _
) As AggregationView(Of TSource,Nullable(Of Integer))
```

C#

```
public static AggregationView<TSource,Nullable<int>> LiveSum<TSource>(
   View<TSource> source,
```

```
    Expression<Func<TSource,Nullable<int>>> selector
)
```

Parameters

source

A view containing the values to calculate the sum of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveSum(View<Decimal>) Method

A view containing the values to calculate the sum of.

Computes the sum of a view of System.Decimal values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveSum( _
   ByVal source As View(Of Decimal) _
) As AggregationView(Of Decimal,Decimal)
```

C#

```
public static AggregationView<decimal,decimal> LiveSum(
   View<decimal> source
)
```

Parameters

source

A view containing the values to calculate the sum of.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveSum(View<Nullable<Decimal>>) Method
A view containing the values to calculate the sum of.

Computes the sum of a view of nullable System.Decimal values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveSum( _
   ByVal source As View(Of Nullable(Of Decimal)) _
) As AggregationView(Of Nullable(Of Decimal),Nullable(Of Decimal))
```

C#

```
public static AggregationView<Nullable<decimal>,Nullable<decimal>> LiveSum(
   View<Nullable<decimal>> source
)
```

Parameters

source

A view containing the values to calculate the sum of.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the entire source collection and aggregate

it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveSum<TSource>(View<TSource>,Expression<Func<TSource,Decimal>>) Method
The type of the elements of *source*.

A view containing the values to calculate the sum of.

A transform function to apply to each element.

Computes the sum of a view of System.Decimal values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveSum(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Decimal)) _
) As AggregationView(Of TSource,Decimal)
```

C#

```
public static AggregationView<TSource,decimal> LiveSum<TSource>(
   View<TSource> source,
   Expression<Func<TSource,decimal>> selector
)
```

Parameters

source

A view containing the values to calculate the sum of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveSum<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Decimal>>>) Method
The type of the elements of *source*.

A view containing the values to calculate the sum of.

A transform function to apply to each element.

Computes the sum of a view of nullable System.Decimal values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveSum(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Decimal))) _
) As AggregationView(Of TSource,Nullable(Of Decimal))
```

C#

```
public static AggregationView<TSource,Nullable<decimal>> LiveSum<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<decimal>>> selector
)
```

Parameters

source

A view containing the values to calculate the sum of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveSum(View<Int64>) Method
A view containing the values to calculate the sum of.

Computes the sum of a view of System.Int64 values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveSum( _
   ByVal source As View(Of Long) _
) As AggregationView(Of Long,Long)
```

C#

```
public static AggregationView<long,long> LiveSum(
   View<long> source
)
```

Parameters

source

A view containing the values to calculate the sum of.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveSum(View<Nullable<Int64>>) Method
A view containing the values to calculate the sum of.

Computes the sum of a view of nullable System.Int64 values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveSum( _
   ByVal source As View(Of Nullable(Of Long)) _
) As AggregationView(Of Nullable(Of Long),Nullable(Of Long))
```

C#

```
public static AggregationView<Nullable<long>,Nullable<long>> LiveSum(
```

```
   View<Nullable<long>> source
)
```

Parameters

source

A view containing the values to calculate the sum of.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveSum<TSource>(View<TSource>,Expression<Func<TSource,Int64>>) Method
The type of the elements of *source*.

A view containing the values to calculate the sum of.

A transform function to apply to each element.

Computes the sum of a view of System.Int64 values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveSum(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Long)) _
) As AggregationView(Of TSource,Long)
```

C#

```
public static AggregationView<TSource,long> LiveSum<TSource>(
   View<TSource> source,
   Expression<Func<TSource,long>> selector
)
```

Parameters

source

A view containing the values to calculate the sum of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveSum<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Int64>>>) Method
The type of the elements of *source*.

A view containing the values to calculate the sum of.

A transform function to apply to each element.

Computes the sum of a view of nullable System.Int64 values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveSum(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Long))) _
) As AggregationView(Of TSource,Nullable(Of Long))
```

C#

```
public static AggregationView<TSource,Nullable<long>> LiveSum<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<long>>> selector
)
```

Parameters

source

A view containing the values to calculate the sum of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveSum(View<Double>) Method
A view containing the values to calculate the sum of.

Computes the sum of a view of System.Double values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveSum( _
   ByVal source As View(Of Double) _
) As AggregationView(Of Double,Double)
```

C#

```
public static AggregationView<double,double> LiveSum(
   View<double> source
)
```

Parameters

source

A view containing the values to calculate the sum of.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveSum(View<Nullable<Double>>) Method

A view containing the values to calculate the sum of.

Computes the sum of a view of nullable System.Double values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveSum( _
   ByVal source As View(Of Nullable(Of Double)) _
) As AggregationView(Of Nullable(Of Double),Nullable(Of Double))
```

C#

```
public static AggregationView<Nullable<double>,Nullable<double>> LiveSum(
   View<Nullable<double>> source
)
```

Parameters

source

A view containing the values to calculate the sum of.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveSum<TSource>(View<TSource>,Expression<Func<TSource,Double>>) Method
The type of the elements of *source*.

A view containing the values to calculate the sum of.

A transform function to apply to each element.

Computes the sum of a view of System.Double values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveSum(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Double)) _
) As AggregationView(Of TSource,Double)
```

C#

```
public static AggregationView<TSource,double> LiveSum<TSource>(
   View<TSource> source,
   Expression<Func<TSource,double>> selector
)
```

Parameters

source

A view containing the values to calculate the sum of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveSum<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Double>>>) Method
The type of the elements of *source*.

A view containing the values to calculate the sum of.

A transform function to apply to each element.

Computes the sum of a view of nullable System.Double values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveSum(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Double))) _
) As AggregationView(Of TSource,Nullable(Of Double))
```

C#

```
public static AggregationView<TSource,Nullable<double>> LiveSum<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<double>>> selector
)
```

Parameters

source

A view containing the values to calculate the sum of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveSum(View<Single>) Method
A view containing the values to calculate the sum of.

Computes the sum of a view of System.Single values.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function LiveSum( _`
`  ByVal source As View(Of Single) _`
`) As AggregationView(Of Single,Single)`

C#

`public static AggregationView<float,float> LiveSum(`
`  View<float> source`
`)`

Parameters

source

A view containing the values to calculate the sum of.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveSum(View<Nullable<Single>>) Method
A view containing the values to calculate the sum of.

Computes the sum of a view of nullable System.Single values.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function LiveSum( _`
`  ByVal source As View(Of Nullable(Of Single)) _`
`) As AggregationView(Of Nullable(Of Single),Nullable(Of Single))`

C#

`public static AggregationView<Nullable<float>,Nullable<float>> LiveSum(`
`  View<Nullable<float>> source`
`)`

Parameters

source

A view containing the values to calculate the sum of.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveSum<TSource>(View<TSource>,Expression<Func<TSource,Single>>) Method
The type of the elements of *source*.

A view containing the values to calculate the sum of.

A transform function to apply to each element.

Computes the sum of a view of System.Single values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)	
`Public Overloads Shared Function LiveSum(Of TSource)( _` `  ByVal source As View(Of TSource), _`	

```
   ByVal selector As Expression(Of Func(Of TSource,Single)) _
) As AggregationView(Of TSource,Single)
```

C#

```
public static AggregationView<TSource,float> LiveSum<TSource>(
   View<TSource> source,
   Expression<Func<TSource,float>> selector
)
```

Parameters

source

A view containing the values to calculate the sum of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveSum<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Single>>>) Method
The type of the elements of *source*.

A view containing the values to calculate the sum of.

A transform function to apply to each element.

Computes the sum of a view of nullable System.Single values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveSum(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Single))) _
) As AggregationView(Of TSource,Nullable(Of Single))
```

C#

```
public static AggregationView<TSource,Nullable<float>> LiveSum<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<float>>> selector
)
```

Parameters

source

A view containing the values to calculate the sum of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

Ordering<T>
The type of the elements of the collection.

Represents a sorted C1.LiveLinq.Indexing.IIndexedSource<T>.

Object Model

Syntax

Visual Basic (Declaration)	
`Public Class Ordering(Of T)` `  Implements C1.LiveLinq.Indexing.IIndexedSource(Of T)`	
C#	
`public class Ordering<T> : C1.LiveLinq.Indexing.IIndexedSource<T>`	

Type Parameters

T

The type of the elements of the collection.

Remarks

An **Ordering<T>** can be obtained as a result of an OrderBy or OrderByDescending query operator. Query operators (extension methods) ThenBy and ThenByDescending operate on objects of type **Ordering<T>**.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.Ordering<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Ordering<T> Members
C1.LiveLinq Namespace

Overview
The type of the elements of the collection.

Represents a sorted C1.LiveLinq.Indexing.IIndexedSource<T>.

Object Model

Syntax

Visual Basic (Declaration)
`Public Class Ordering(Of T)` `  Implements C1.LiveLinq.Indexing.IIndexedSource(Of T)`
C#
`public class Ordering<T> : C1.LiveLinq.Indexing.IIndexedSource<T>`

Type Parameters

T

The type of the elements of the collection.

Remarks

An **Ordering<T>** can be obtained as a result of an OrderBy or OrderByDescending query operator. Query operators (extension methods) ThenBy and ThenByDescending operate on objects of type **Ordering<T>**.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.Ordering<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Ordering<T> Members
C1.LiveLinq Namespace

Members
Methods

The following tables list the members exposed by Ordering<T>.

Public Methods

	Name	Description
	GetEnumerator	Returns an enumerator that iterates through the Ordering<T>.
	ThenBy<TKey>	Performs a subsequent ordering of the elements in a collection in ascending order according to a key.
	ThenByDescending<TKey>	Performs a subsequent ordering of the elements in a collection in descending order according to a key.

Top

See Also

Reference

Ordering<T> Class
C1.LiveLinq Namespace

Methods
For a list of all members of this type, see Ordering<T> members.

Public Methods

	Name	Description
	GetEnumerator	Returns an enumerator that iterates through the Ordering<T>.
	ThenBy<TKey>	Performs a subsequent ordering of the elements in a collection in ascending order according to a key.

	ThenByDescending<TKey>	Performs a subsequent ordering of the elements in a collection in descending order according to a key.

Top

See Also

Reference

Ordering<T> Class
C1.LiveLinq Namespace

GetEnumerator Method
Returns an enumerator that iterates through the Ordering<T>.

Syntax

Visual Basic (Declaration)
`Public Function GetEnumerator() As IEnumerator(Of T)`
C#
`public IEnumerator<T> GetEnumerator()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Ordering<T> Class
Ordering<T> Members

ThenBy<TKey> Method
The type of the key returned by *keySelector*.

A function to extract a key from each element.

Performs a subsequent ordering of the elements in a collection in ascending order according to a key.

Syntax

Visual Basic (Declaration)

`Public Function ThenBy(Of TKey)( _`
`  ByVal keySelector As Expression(Of Func(Of T,TKey)) _`
`) As Ordering(Of T)`

C#

`public Ordering<T> ThenBy<TKey>(`
`  Expression<Func<T,TKey>> keySelector`
`)`

Parameters

keySelector

A function to extract a key from each element.

Type Parameters

TKey

The type of the key returned by *keySelector*.

Return Value

A collection whose elements are sorted according to a key.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Ordering<T> Class
Ordering<T> Members

ThenByDescending<TKey> Method
The type of the key returned by *keySelector*.

A function to extract a key from each element.

Performs a subsequent ordering of the elements in a collection in descending order according to a key.

Syntax

Visual Basic (Declaration)

`Public Function ThenByDescending(Of TKey)( _`
`  ByVal keySelector As Expression(Of Func(Of T,TKey)) _`
`) As Ordering(Of T)`

C#

`public Ordering<T> ThenByDescending<TKey>(`
`  Expression<Func<T,TKey>> keySelector`
`)`

Parameters

keySelector

A function to extract a key from each element.

Type Parameters

TKey

The type of the key returned by *keySelector*.

Return Value

A collection whose elements are sorted in descending order according to a key.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

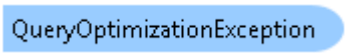
See Also

Reference

Ordering<T> Class
Ordering<T> Members

QueryOptimizationException
Represents an exception that is thrown when Hints in a query require using a certain mandatory optimization which is impossible in the current query context.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class QueryOptimizationException` `  Inherits System.Exception`
C#
`public class QueryOptimizationException : System.Exception`

Inheritance Hierarchy

System.Object
  System.Exception
    C1.LiveLinq.QueryOptimizationException

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

QueryOptimizationException Members
C1.LiveLinq Namespace

Overview
Represents an exception that is thrown when Hints in a query require using a certain mandatory optimization which is impossible in the current query context.

Object Model

QueryOptimizationException

Syntax

Visual Basic (Declaration)
`Public Class QueryOptimizationException` `  Inherits System.Exception`
C#
`public class QueryOptimizationException : System.Exception`

Inheritance Hierarchy

System.Object
  System.Exception
    C1.LiveLinq.QueryOptimizationException

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

QueryOptimizationException Members
C1.LiveLinq Namespace

Members
Properties Methods

The following tables list the members exposed by QueryOptimizationException.

Public Constructors

	Name	Description

	QueryOptimizationException Constructor	Overloaded.

Top

Public Properties

	Name	Description
	Data	(Inherited from System.Exception)
	HelpLink	(Inherited from System.Exception)
	InnerException	(Inherited from System.Exception)
	Message	(Inherited from System.Exception)
	Source	(Inherited from System.Exception)
	StackTrace	(Inherited from System.Exception)
	TargetSite	(Inherited from System.Exception)

Top

Public Methods

	Name	Description
	GetBaseException	(Inherited from System.Exception)
	GetObjectData	(Inherited from System.Exception)
	GetType	(Inherited from System.Exception)
	ToString	(Inherited from System.Exception)

Top

See Also

Reference

QueryOptimizationException Class
C1.LiveLinq Namespace

QueryOptimizationException Constructor

Overload List

Overload	Description
QueryOptimizationException Constructor()	Initializes a new instance of the QueryOptimizationException class. This is the default constructor.
QueryOptimizationException Constructor(String)	Initializes a new instance of the QueryOptimizationException class with the specified string.
QueryOptimizationException Constructor(String,Exception)	Initializes a new instance of the QueryOptimizationException class using the specified string and inner exception.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

QueryOptimizationException Class
QueryOptimizationException Members

QueryOptimizationException Constructor()
Initializes a new instance of the QueryOptimizationException class. This is the default constructor.

Syntax

Visual Basic (Declaration)

```
Public Function New()
```

C#

```
public QueryOptimizationException()
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

QueryOptimizationException Class
QueryOptimizationException Members
Overload List

QueryOptimizationException Constructor(String)
The string to display when the exception is thrown.

Initializes a new instance of the QueryOptimizationException class with the specified string.

Syntax

Visual Basic (Declaration)

```
Public Function New( _
   ByVal message As String _
)
```

C#

```
public QueryOptimizationException(
   string message
)
```

Parameters

message

The string to display when the exception is thrown.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

QueryOptimizationException Class
QueryOptimizationException Members
Overload List

QueryOptimizationException Constructor(String,Exception)
The string to display when the exception is thrown.

Gets the **Exception** instance that caused the current exception.

Initializes a new instance of the QueryOptimizationException class using the specified string and inner exception.

Syntax

Visual Basic (Declaration)

```
Public Function New( _
   ByVal message As String, _
   ByVal inner As Exception _
)
```

C#

```
public QueryOptimizationException(
   string message,
   Exception inner
)
```

Parameters

message

The string to display when the exception is thrown.

inner

Gets the **Exception** instance that caused the current exception.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

QueryOptimizationException Class
QueryOptimizationException Members
Overload List

SourceChangeEventArgs<T>
Type of the changed object.

Provides data for the IObservableSource<T>.Changed event.

Object Model

Syntax

Visual Basic (Declaration)
`Public Class SourceChangeEventArgs(Of T)` `  Inherits System.EventArgs`
C#
`public class SourceChangeEventArgs<T> : System.EventArgs`

Type Parameters

T

Type of the changed object.

Inheritance Hierarchy

System.Object
  System.EventArgs
    C1.LiveLinq.SourceChangeEventArgs<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SourceChangeEventArgs<T> Members
C1.LiveLinq Namespace

Overview
Type of the changed object.

Provides data for the IObservableSource<T>.Changed event.

Object Model

Syntax

Visual Basic (Declaration)
`Public Class SourceChangeEventArgs(Of T)` `  Inherits System.EventArgs`
C#
`public class SourceChangeEventArgs<T> : System.EventArgs`

Type Parameters

T

Type of the changed object.

Inheritance Hierarchy

System.Object
  System.EventArgs
    C1.LiveLinq.SourceChangeEventArgs<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SourceChangeEventArgs<T> Members
C1.LiveLinq Namespace

Members
Properties Methods

The following tables list the members exposed by SourceChangeEventArgs<T>.

Public Constructors

	Name	Description
	SourceChangeEventArgs<T> Constructor	Initializes a new instance of the SourceChangeEventArgs<T> class.

Top

Public Properties

	Name	Description
	ChangeType	Gets the type of change.
	Item	Gets the object that is being changed.

	Ordinal	Gets the ordinal position of the collection item that is being changed.

Top

Public Methods

	Name	Description
	Equals	Overloaded. Determines whether the specified object is equal to the current object.
	GetHashCode	Return a hash code for this instance.
	ToString	Returns a string that represents this instance of SourceChangeEventArgs<T>.

Top

See Also

Reference

SourceChangeEventArgs<T> Class
C1.LiveLinq Namespace

SourceChangeEventArgs<T> Constructor
Changed object.

Type of change.

Ordinal position of the changed item.

Initializes a new instance of the SourceChangeEventArgs<T> class.

Syntax

Visual Basic (Declaration)

```
Public Function New( _
   ByVal item As T, _
   ByVal changeType As SourceChangeType, _
```

```
   ByVal ordinal As Integer _
)
```

C#

```
public SourceChangeEventArgs<T>(
   T item,
   SourceChangeType changeType,
   int ordinal
)
```

Parameters

item

Changed object.

changeType

Type of change.

ordinal

Ordinal position of the changed item.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SourceChangeEventArgs<T> Class
SourceChangeEventArgs<T> Members

Methods

For a list of all members of this type, see SourceChangeEventArgs<T> members.

Public Methods

	Name	Description

	Equals	Overloaded. Determines whether the specified object is equal to the current object.
	GetHashCode	Return a hash code for this instance.
	ToString	Returns a string that represents this instance of SourceChangeEventArgs<T>.

Top

See Also

Reference

SourceChangeEventArgs<T> Class
C1.LiveLinq Namespace

Equals Method
Determines whether the specified object is equal to the current object.

Overload List

Overload	Description
Equals(Object)	Determines whether the specified object is equal to the current object.
Equals(SourceChangeEventArgs<T>)	Determines whether the specified object is equal to the current object.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SourceChangeEventArgs<T> Class
SourceChangeEventArgs<T> Members

Equals(Object) Method
The object to compare with the current object.

Determines whether the specified object is equal to the current object.

Syntax

Visual Basic (Declaration)

```
Public Overloads Overrides Function Equals( _
   ByVal obj As Object _
) As Boolean
```

C#

```
public override bool Equals(
   object obj
)
```

Parameters

obj

The object to compare with the current object.

Return Value

true if the specified object is equal to the current one; otherwise, false.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SourceChangeEventArgs<T> Class
SourceChangeEventArgs<T> Members
Overload List

Equals(SourceChangeEventArgs<T>) Method

The object to compare with the current object.

Determines whether the specified object is equal to the current object.

Syntax

Visual Basic (Declaration)	

`Public Overloads Function Equals( _`
`  ByVal args As SourceChangeEventArgs(Of T) _`
`) As Boolean`	

C#	

`public bool Equals(`
`  SourceChangeEventArgs<T> args`
`)`	

Parameters

args

The object to compare with the current object.

Return Value

true if the specified object is equal to the current one; otherwise, false.

Remarks

Two SourceChangeEventArgs<T> are considered equal if the values of their Item and ChangeType properties are equal.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SourceChangeEventArgs<T> Class
SourceChangeEventArgs<T> Members
Overload List

GetHashCode Method
Return a hash code for this instance.

Syntax

Visual Basic (Declaration)
`Public Overrides Function GetHashCode() As Integer`
C#
`public override int GetHashCode()`

Return Value

A 32-bit signed integer hash code.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SourceChangeEventArgs<T> Class
SourceChangeEventArgs<T> Members

ToString Method
Returns a string that represents this instance of SourceChangeEventArgs<T>.

Syntax

Visual Basic (Declaration)
`Public Overrides Function ToString() As String`
C#

```
public override string ToString()
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SourceChangeEventArgs<T> Class
SourceChangeEventArgs<T> Members

Properties
For a list of all members of this type, see SourceChangeEventArgs<T> members.

Public Properties

	Name	Description
	ChangeType	Gets the type of change.
	Item	Gets the object that is being changed.
	Ordinal	Gets the ordinal position of the collection item that is being changed.

Top

See Also

Reference

SourceChangeEventArgs<T> Class
C1.LiveLinq Namespace

ChangeType Property
Gets the type of change.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property ChangeType As SourceChangeType`
C#
`public SourceChangeType ChangeType {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SourceChangeEventArgs<T> Class
SourceChangeEventArgs<T> Members

Item Property
Gets the object that is being changed.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Item As T`
C#
`public T Item {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SourceChangeEventArgs<T> Class
SourceChangeEventArgs<T> Members

Ordinal Property
Gets the ordinal position of the collection item that is being changed.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Ordinal As Integer`
C#
`public int Ordinal {get;}`

Remarks

This property can return -1 (ordinal unknown) if the collection cannot provide this information (if IObservableSource<T>.SupportsItemOrdinals returns **false**).

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SourceChangeEventArgs<T> Class
SourceChangeEventArgs<T> Members

Enumerations

IndexingHintAction
Specifies the actions taken by LiveLinq query optimizer when it encounters an **Indexed()** hint applied to an expression (usually, a property) in a query.

Syntax

Visual Basic (Declaration)

```
Public Enum IndexingHintAction
   Inherits System.Enum
```

C#

```
public enum IndexingHintAction : System.Enum
```

Members

Member	Description
Mandatory	Create an index for this expression, if such index does not already exist. Use this index in executing the query (perform an index scan using that index). Throw an exception if it is impossible to use it in query execution.
Optional	Create an index for this expression, if such index does not already exist. Use this index in executing the query, if it is possible. Do not throw exception if it is impossible to use that index in query execution.
UseExistingIndex	Check that there exists an index for this expression. If it does not exist, throw an exception. Use this index in executing the query. Throw an exception if it is impossible to use this index in query execution.

Inheritance Hierarchy

System.Object
  System.ValueType
    System.Enum
      C1.LiveLinq.IndexingHintAction

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1.LiveLinq Namespace

Order

Indicates if a certain order is required in the result collection of an operation.

Syntax

Visual Basic (Declaration)
`Public Enum Order` `  Inherits System.Enum`
C#
`public enum Order : System.Enum`

Members

Member	Description
Ascending	The resulting collection must be ordered in ascending key order.
Descending	The resulting collection must be ordered in descending key order.
Unordered	No particular order is required in the resulting collection.

Inheritance Hierarchy

System.Object
  System.ValueType
    System.Enum
      C1.LiveLinq.Order

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1.LiveLinq Namespace

SourceChangeType

Describes a change occurring in a collection.

Syntax

Visual Basic (Declaration)
`Public Enum SourceChangeType` `  Inherits System.Enum`
C#
`public enum SourceChangeType : System.Enum`

Members

Member	Description
Add	An item is added to the collection.
Modify	At least one of the properties of an item has changed its value.
Remove	An item is removed from the collection.
Reset	Multiple items have changed in the collection. Indexes must be rebuilt.

Inheritance Hierarchy

System.Object
  System.ValueType
    System.Enum
      C1.LiveLinq.SourceChangeType

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1.LiveLinq Namespace

TransactionState
Enumeration of the possible states an ITransaction can be in.

Syntax

Visual Basic (Declaration)
`Public Enum TransactionState` `  Inherits System.Enum`
C#
`public enum TransactionState : System.Enum`

Members

Member	Description
Committed	A transaction is committed.
Committing	A transaction is in the process of being committed.
Open	A transaction is open.
RolledBack	A transaction is rolled back.
RollingBack	A transaction is in the process of being rolled back.

Inheritance Hierarchy

System.Object
  System.ValueType
    System.Enum
      C1.LiveLinq.TransactionState

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

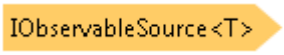
C1.LiveLinq Namespace

Interfaces

IObservableSource<T>
The type of the elements in the collection.

Provides methods and events that are required for LiveLinq functionality, indexing and live views.

Object Model



Syntax

Visual Basic (Declaration)
`Public Interface IObservableSource(Of T)`
C#
`public interface IObservableSource<T>`

Type Parameters

T

The type of the elements in the collection.

Remarks

Indexing and live view functionality is available for any collection that supports change notifications necessary for maintaining indexes and live views, that is, fires events when an item is added to or removed from the collection and when a property of the item changes. So, the members of this interface are mostly concerned with providing such notifications.

Classes implementing this interface usually also implement C1.LiveLinq.Indexing.IIndexedSource<T>.

Both these interfaces are implemented by all main LiveLinq collection classes: C1.LiveLinq.Collections.IndexedCollection<T>, C1.LiveLinq.AdoNet.IndexedDataTable<TRow>, C1.LiveLinq.LiveViews.View<T>.

You need to implement this interface only if you want to define your own indexable collection classes and then only if they don't inherit from C1.LiveLinq.Collections.IndexedCollection<T>, see LiveLinq to Objects: IndexedCollection(T) and other collection classes.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

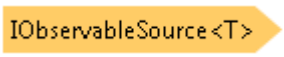
Reference

IObservableSource<T> Members
C1.LiveLinq Namespace

Overview
The type of the elements in the collection.

Provides methods and events that are required for LiveLinq functionality, indexing and live views.

Object Model



Syntax

Visual Basic (Declaration)
`Public Interface IObservableSource(Of T)`
C#
`public interface IObservableSource<T>`

Type Parameters

T

The type of the elements in the collection.

Remarks

Indexing and live view functionality is available for any collection that supports change notifications necessary for maintaining indexes and live views, that is, fires events when an item is added to or removed from the collection and when a property of the item changes. So, the members of this interface are mostly concerned with providing such notifications.

Classes implementing this interface usually also implement C1.LiveLinq.Indexing.IIndexedSource<T>.

Both these interfaces are implemented by all main LiveLinq collection classes: C1.LiveLinq.Collections.IndexedCollection<T>, C1.LiveLinq.AdoNet.IndexedDataTable<TRow>, C1.LiveLinq.LiveViews.View<T>.

You need to implement this interface only if you want to define your own indexable collection classes and then only if they don't inherit from C1.LiveLinq.Collections.IndexedCollection<T>, see LiveLinq to Objects: IndexedCollection(T) and other collection classes.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IObservableSource<T> Members
C1.LiveLinq Namespace

Members
Properties Methods Events

The following tables list the members exposed by IObservableSource<T>.

Public Properties

	Name	Description

	CreateNew	This delegate is used to create new items. If it is null, a public parameterless constructor of type **T** is used.
	IsDeletedStateAvailable	Gets a value indicating whether an item of this collection can still return correct property values after it has been deleted from the collection.
	SupportsItemOrdinals	Gets a value that indicates whether this collection is capable of providing the ordinal position of the changed item when it notifies of an item change.

Top

Public Methods

	Name	Description
	EnableItemOrdinals	After this method is called, the collection is required to provide SourceChangeEventArgs<T>.Ordinal in event data.

Top

Public Events

	Name	Description
	Changed	Occurs after an item of the collection or the entire collection has changed.

Top

See Also

Reference

IObservableSource<T> Interface
C1.LiveLinq Namespace

Methods
For a list of all members of this type, see IObservableSource<T> members.

Public Methods

	Name	Description
	EnableItemOrdinals	After this method is called, the collection is required to provide SourceChangeEventArgs<T>.Ordinal in event data.

Top

See Also

Reference

IObservableSource<T> Interface
C1.LiveLinq Namespace

EnableItemOrdinals Method
After this method is called, the collection is required to provide SourceChangeEventArgs<T>.Ordinal in event data.

Syntax

Visual Basic (Declaration)
`Sub EnableItemOrdinals()`
C#
`void EnableItemOrdinals()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IObservableSource<T> Interface
IObservableSource<T> Members

Properties
For a list of all members of this type, see IObservableSource<T> members.

Public Properties

	Name	Description
	CreateNew	This delegate is used to create new items. If it is null, a public parameterless constructor of type **T** is used.
	IsDeletedStateAvailable	Gets a value indicating whether an item of this collection can still return correct property values after it has been deleted from the collection.
	SupportsItemOrdinals	Gets a value that indicates whether this collection is capable of providing the ordinal position of the changed item when it notifies of an item change.

Top

See Also

Reference

IObservableSource<T> Interface
C1.LiveLinq Namespace

CreateNew Property
This delegate is used to create new items. If it is null, a public parameterless constructor of type **T** is used.

Syntax

Visual Basic (Declaration)

```
ReadOnly Property CreateNew As Func(Of T)
```

C#

```
Func<T> CreateNew {get;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IObservableSource<T> Interface
IObservableSource<T> Members

IsDeletedStateAvailable Property
Gets a value indicating whether an item of this collection can still return correct property values after it has been deleted from the collection.

Syntax

Visual Basic (Declaration)
`ReadOnly Property IsDeletedStateAvailable As Boolean`
C#
`bool IsDeletedStateAvailable {get;}`

Property Value

true if deleted items can still be used to get property values.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IObservableSource<T> Interface
IObservableSource<T> Members
DeletedStateIsAvailableAttribute Class

SupportsItemOrdinals Property

Gets a value that indicates whether this collection is capable of providing the ordinal position of the changed item when it notifies of an item change.

Syntax

Visual Basic (Declaration)
`ReadOnly Property SupportsItemOrdinals As Boolean`
C#
`bool SupportsItemOrdinals {get;}`

Property Value

true if the collection can provide item ordinal positions.

Remarks

If this property returns **true**, LiveLinq can call the EnableItemOrdinals method to require providing ordinals.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IObservableSource<T> Interface
IObservableSource<T> Members

Events

>

Name	Description
Changed	Occurs after an item of the collection or the entire collection has changed.

Top

See Also

Reference

IObservableSource<T> Interface
C1.LiveLinq Namespace

Changed Event
Occurs after an item of the collection or the entire collection has changed.

Syntax

Visual Basic (Declaration)
`Event Changed As EventHandler(Of SourceChangeEventArgs(Of T))`
C#
`event EventHandler<SourceChangeEventArgs<T>> Changed`

Event Data

The event handler receives an argument of type SourceChangeEventArgs<T> containing data related to this event. The following **SourceChangeEventArgs<T>** properties provide information specific to this event.

Property	Description
ChangeType	Gets the type of change.
Item	Gets the object that is being changed.
Ordinal	Gets the ordinal position of the collection item that is being changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

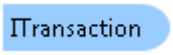
See Also

Reference

IObservableSource<T> Interface
IObservableSource<T> Members

ITransaction
Represents a transaction with an explicit scope.

Object Model



Syntax

Visual Basic (Declaration)
`Public Interface ITransaction`
C#
`public interface ITransaction`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

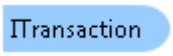
See Also

Reference

ITransaction Members
C1.LiveLinq Namespace

Overview
Represents a transaction with an explicit scope.

Object Model



Syntax

Visual Basic (Declaration)	
`Public Interface ITransaction`	
C#	
`public interface ITransaction`	

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ITransaction Members
C1.LiveLinq Namespace

Members
Properties Methods

The following tables list the members exposed by ITransaction.

Public Properties

	Name	Description
	HasChanges	Gets a value indicating whether any changes were made in the scope of this transaction.
	State	Gets the current state of the transaction.

Top

Public Methods

	Name	Description
	Commit	Commits changes that were made while this transaction's scope was open.

	Rollback	Rolls back changes that were made while this transaction's scope was open.
	Scope	Opens a transaction scope.

Top

See Also

Reference

ITransaction Interface
C1.LiveLinq Namespace

Methods
For a list of all members of this type, see ITransaction members.

Public Methods

	Name	Description
	Commit	Commits changes that were made while this transaction's scope was open.
	Rollback	Rolls back changes that were made while this transaction's scope was open.
	Scope	Opens a transaction scope.

Top

See Also

Reference

ITransaction Interface
C1.LiveLinq Namespace

Commit Method
Commits changes that were made while this transaction's scope was open.

Syntax

Visual Basic (Declaration)	

```
Sub Commit()
```

C#

```
void Commit()
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ITransaction Interface
ITransaction Members

Rollback Method
Rolls back changes that were made while this transaction's scope was open.

Syntax

Visual Basic (Declaration)

```
Sub Rollback()
```

C#

```
void Rollback()
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ITransaction Interface
ITransaction Members

Scope Method

Opens a transaction scope.

Syntax

Visual Basic (Declaration)
`Function Scope() As IDisposable`
C#
`IDisposable Scope()`

Return Value

An instance of System.IDisposable that will close the scope when its System.IDisposable.Dispose method is called.

Remarks

The transaction tracks changes only when they are made inside an open scope.

Calling System.IDisposable.Dispose on the return value closes the scope.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ITransaction Interface
ITransaction Members

Properties

For a list of all members of this type, see ITransaction members.

Public Properties

	Name	Description

	HasChanges	Gets a value indicating whether any changes were made in the scope of this transaction.
	State	Gets the current state of the transaction.

Top

See Also

Reference

ITransaction Interface
C1.LiveLinq Namespace

HasChanges Property
Gets a value indicating whether any changes were made in the scope of this transaction.

Syntax

Visual Basic (Declaration)
`ReadOnly Property HasChanges As Boolean`
C#
`bool HasChanges {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ITransaction Interface
ITransaction Members

State Property
Gets the current state of the transaction.

Syntax

Visual Basic (Declaration)
`ReadOnly Property State As TransactionState`
C#
`TransactionState State {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ITransaction Interface
ITransaction Members

C1.LiveLinq.AdoNet Namespace

Overview

Classes

	Class	Description
	AdoNetExtensions	Provides a set of static (extension) methods for LiveLinq to DataSet.
	IndexedDataTable<TRow>	A wrapper for the standard ADO.NET System.Data.DataTable class allowing to use ADO.NET data sources in LiveLinq indexing and live views.

See Also

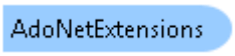
Reference

C1.LiveLinq.4 Assembly

Classes

AdoNetExtensions
Provides a set of static (extension) methods for LiveLinq to DataSet.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit NotInheritable Class AdoNetExtensions`
C#
`public static class AdoNetExtensions`

Inheritance Hierarchy

System.Object
  C1.LiveLinq.AdoNet.AdoNetExtensions

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

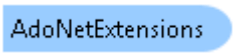
See Also

Reference

AdoNetExtensions Members
C1.LiveLinq.AdoNet Namespace

Overview
Provides a set of static (extension) methods for LiveLinq to DataSet.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit NotInheritable Class AdoNetExtensions`
C#
`public static class AdoNetExtensions`

Inheritance Hierarchy

System.Object
  C1.LiveLinq.AdoNet.AdoNetExtensions

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

AdoNetExtensions Members
C1.LiveLinq.AdoNet Namespace

Members
Methods

The following tables list the members exposed by AdoNetExtensions.

Public Methods

	Name	Description
S	AsIndexed	Overloaded. Wraps a System.Data.DataTable in an IndexedDataTable<DataRow> so it can be indexed and queried using the optimized query operators from C1.LiveLinq.IndexedQueryExtensions.
S	AsLive	Overloaded. Creates a view based on the specified ADO.NET

		System.Data.DataTable.
S	BeginUpdate	Indicates that massive changes are being made in code, until EndUpdate, to data tables in this System.Data.DataSet.
S	EndUpdate	Indicates the end of massive changes to data tables in this System.Data.DataSet started with BeginUpdate.
S	IndexedField	Overloaded. A hint to create and use an index on the specified data column. The hint has default action.

Top

See Also

Reference

AdoNetExtensions Class
C1.LiveLinq.AdoNet Namespace

Methods
For a list of all members of this type, see AdoNetExtensions members.

Public Methods

	Name	Description
S	AsIndexed	Overloaded. Wraps a System.Data.DataTable in an IndexedDataTable<DataRow> so it can be indexed and queried using the optimized query operators from C1.LiveLinq.IndexedQueryExtensions.
S	AsLive	Overloaded. Creates a view based on the specified ADO.NET System.Data.DataTable.
S	BeginUpdate	Indicates that massive changes are being made in code, until EndUpdate, to data tables in this System.Data.DataSet.
S	EndUpdate	Indicates the end of massive changes to data tables in this

		System.Data.DataSet started with BeginUpdate.
S	IndexedField	Overloaded. A hint to create and use an index on the specified data column. The hint has default action.

Top

See Also

Reference

AdoNetExtensions Class
C1.LiveLinq.AdoNet Namespace

AsIndexed Method
Wraps a System.Data.DataTable in an IndexedDataTable<DataRow> so it can be indexed and queried using the optimized query operators from C1.LiveLinq.IndexedQueryExtensions.

Overload List

Overload	Description
AsIndexed(DataTable)	Wraps a System.Data.DataTable in an IndexedDataTable<DataRow> so it can be indexed and queried using the optimized query operators from C1.LiveLinq.IndexedQueryExtensions.
AsIndexed<TRow>(DataTable)	Wraps a System.Data.DataTable in an IndexedDataTable<TRow> so it can be indexed and queried using the optimized query operators from C1.LiveLinq.IndexedQueryExtensions.
AsIndexed<TRow>(TypedTableBase<TRow>)	Wraps a typed ADO.NET data table in an IndexedDataTable<TRow> so it can be indexed and queried using the optimized query operators from C1.LiveLinq.IndexedQueryExtensions.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

AdoNetExtensions Class
AdoNetExtensions Members

AsIndexed(DataTable) Method
A System.Data.DataTable to represent as an IndexedDataTable<DataRow>.

Wraps a System.Data.DataTable in an IndexedDataTable<DataRow> so it can be indexed and queried using the optimized query operators from C1.LiveLinq.IndexedQueryExtensions.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function AsIndexed( _
   ByVal table As DataTable _
) As IndexedDataTable(Of DataRow)
```

C#

```
public static IndexedDataTable<DataRow> AsIndexed(
   DataTable table
)
```

Parameters

table

A System.Data.DataTable to represent as an IndexedDataTable<DataRow>.

Return Value

An IndexedDataTable<DataRow> that contains the same rows as *table* and enables indexing of its rows.

Remarks

Use this method to index ADO.NET data tables and query them using the query operators optimized with indexing.

Elements of the source data table aren't duplicated or copied to a new collection. This method just wraps the original data table in an IndexedDataTable<DataRow>.

Note: The IndexedDataTable<DataRow> wrapper is owned by the original System.Data.DataTable object (in fact, it is stored in its **ExtendedProperties**). So, if you create a wrapper for the same data table several times, it will be the same object.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

AdoNetExtensions Class
AdoNetExtensions Members
Overload List

AsIndexed<TRow>(DataTable) Method
The type of the rows in the *table*.

A System.Data.DataTable to represent as an IndexedDataTable<TRow>.

Wraps a System.Data.DataTable in an IndexedDataTable<TRow> so it can be indexed and queried using the optimized query operators from C1.LiveLinq.IndexedQueryExtensions.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function AsIndexed(Of TRow As DataRow)( _
   ByVal table As DataTable _
) As IndexedDataTable(Of TRow)
```

C#

```
public static IndexedDataTable<TRow> AsIndexed<TRow>(
   DataTable table
```

```
)
where TRow: DataRow
```

Parameters

table

A System.Data.DataTable to represent as an IndexedDataTable<TRow>.

Type Parameters

TRow

The type of the rows in the *table*.

Return Value

An IndexedDataTable<TRow> that contains the same rows as *table* and enables indexing of its rows.

Remarks

Use this method to index ADO.NET data tables and query them using the query operators optimized with indexing.

Elements of the source data table aren't duplicated or copied to a new collection. This method just wraps the original data table in an IndexedDataTable<TRow>.

Note: The IndexedDataTable<TRow> wrapper is owned by the original System.Data.DataTable object (in fact, it is stored in its **ExtendedProperties**). So, if you create a wrapper for the same data table several times, it will be the same object.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

AdoNetExtensions Class
AdoNetExtensions Members
Overload List

The type of the rows in the *table*.

A typed data table to represent as an IndexedDataTable<TRow>.

Wraps a typed ADO.NET data table in an IndexedDataTable<TRow> so it can be indexed and queried using the optimized query operators from C1.LiveLinq.IndexedQueryExtensions.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function AsIndexed(Of TRow As DataRow)( _
   ByVal table As TypedTableBase(Of TRow) _
) As IndexedDataTable(Of TRow)
```

C#

```
public static IndexedDataTable<TRow> AsIndexed<TRow>(
   TypedTableBase<TRow> table
)
where TRow: DataRow
```

Parameters

table

A typed data table to represent as an IndexedDataTable<TRow>.

Type Parameters

TRow

The type of the rows in the *table*.

Return Value

An IndexedDataTable<TRow> that contains the same rows as *table* and enables indexing of its rows.

Remarks

Use this method to index typed data tables and query them using the query operators optimized with indexing.

Elements of the source data table aren't duplicated or copied to a new collection. This method just wraps the original data table in an IndexedDataTable<TRow>.

Note: The IndexedDataTable<TRow> wrapper is owned by the original System.Data.DataTable object (in fact, it is stored in its **ExtendedProperties**). So, if you create a wrapper for the same data table several times, it will be the same object.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

AdoNetExtensions Class
AdoNetExtensions Members
Overload List

AsLive Method

Creates a view based on the specified ADO.NET System.Data.DataTable.

Overload List

Overload	Description
AsLive(DataTable)	Creates a view based on the specified ADO.NET System.Data.DataTable.
AsLive<TRow>(DataTable)	Creates a view based on the specified ADO.NET System.Data.DataTable.
AsLive<TRow>(TypedTableBase<TRow>)	Creates a view based on the specified typed data table.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

AdoNetExtensions Class
AdoNetExtensions Members

AsLive(DataTable) Method
The System.Data.DataTable to expose as a view.

Creates a view based on the specified ADO.NET System.Data.DataTable.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function AsLive( _
   ByVal table As DataTable _
) As View(Of DataRow)
```

C#

```
public static View<DataRow> AsLive(
   DataTable table
)
```

Parameters

table

The System.Data.DataTable to expose as a view.

Return Value

A view that contains the same elements as *table*.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

AdoNetExtensions Class
AdoNetExtensions Members
Overload List

AsLive<TRow>(DataTable) Method
The type of the rows in the *table*.

The typed data table to expose as a view.

Creates a view based on the specified ADO.NET System.Data.DataTable.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function AsLive(Of TRow As DataRow)( _`
`  ByVal table As DataTable _`
`) As View(Of TRow)`

C#

`public static View<TRow> AsLive<TRow>(`
`  DataTable table`
`)`
`where TRow: DataRow`

Parameters

table

The typed data table to expose as a view.

Type Parameters

TRow

The type of the rows in the *table*.

Return Value

A view that contains the same elements as *table*.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

AdoNetExtensions Class
AdoNetExtensions Members
Overload List

AsLive<TRow>(TypedTableBase<TRow>) Method
The type of the rows in the *table*.

The typed data table to expose as a view.

Creates a view based on the specified typed data table.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function AsLive(Of TRow As DataRow)( _
   ByVal table As TypedTableBase(Of TRow) _
) As View(Of TRow)
```

C#

```
public static View<TRow> AsLive<TRow>(
   TypedTableBase<TRow> table
)
where TRow: DataRow
```

Parameters

table

The typed data table to expose as a view.

Type Parameters

TRow

The type of the rows in the *table*.

Return Value

A view that contains the same elements as *table*.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

AdoNetExtensions Class
AdoNetExtensions Members
Overload List

BeginUpdate Method

The System.Data.DataSet to start massive changes for.

Indicates that massive changes are being made in code, until EndUpdate, to data tables in this System.Data.DataSet.

Syntax

Visual Basic (Declaration)

`Public Shared Sub BeginUpdate( _`
`  ByVal dataSet As DataSet _`
`)`

C#

`public static void BeginUpdate(`
`  DataSet dataSet`
`)`

Parameters

dataSet

The System.Data.DataSet to start massive changes for.

Remarks

This extension method calls BeginUpdate for all data tables of this System.Data.DataSet.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

AdoNetExtensions Class
AdoNetExtensions Members

EndUpdate Method
The System.Data.DataSet where massive changes have ended.

Indicates the end of massive changes to data tables in this System.Data.DataSet started with BeginUpdate.

Syntax

Visual Basic (Declaration)

```
Public Shared Sub EndUpdate( _
   ByVal dataSet As DataSet _
)
```

C#

```
public static void EndUpdate(
   DataSet dataSet
)
```

Parameters

dataSet

The System.Data.DataSet where massive changes have ended.

Remarks

This extension method calls IndexedDataTable<TRow>.EndUpdate for all data tables of this System.Data.DataSet.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

AdoNetExtensions Class
AdoNetExtensions Members

IndexedField Method
A hint to create and use an index on the specified data column. The hint has default action.

Overload List

Overload	Description
IndexedField<T>(DataRow,Int32)	A hint to create and use an index on the specified data column. The hint has default action.
IndexedField<T>(DataRow,DataColumn)	A hint to create and use an index on the specified data column. The hint has default action.
IndexedField<T>(DataRow,String)	A hint to create and use an index on the specified data column. The hint has default action.
IndexedField<T>(DataRow,Int32,IndexingHintAction)	A hint to create and use an index on the specified data column. The hint has specified action.
IndexedField<T>(DataRow,DataColumn,IndexingHintAction)	A hint to create and use an index on the specified data column. The hint

	has specified action.
IndexedField<T>(DataRow,String,IndexingHintAction)	A hint to create and use an index on the specified data column. The hint has specified action.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

AdoNetExtensions Class
AdoNetExtensions Members

IndexedField<T>(DataRow,Int32) Method
Example

The type of the data column

The data row.

The zero-based ordinal position of the column.

A hint to create and use an index on the specified data column. The hint has default action.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function IndexedField(Of T)( _
   ByVal row As DataRow, _
   ByVal columnIndex As Integer _
) As T
```

C#

```
public static T IndexedField<T>(
   DataRow row,
```

```
    int columnIndex
)
```

Parameters

row

The data row.

columnIndex

The zero-based ordinal position of the column.

Type Parameters

T

The type of the data column

Return Value

The data column value.

Remarks

Hints are used declaratively. They tell LiveLinq query optimizer to create and use an index on that column, if possible. When the query is executed, the hint method **IndexedField** is replaced with the standard LINQ to DataSet extension method Field. See C1.LiveLinq.Hints for more details.

Example

- C#

```
var query = from c in customersTable
    where c.IndexedField<string>(0) == "ALFKI"
    select c;
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

AdoNetExtensions Class
AdoNetExtensions Members
Overload List

IndexedField<T>(DataRow,DataColumn) Method
Example

The type of the data column

The data row.

The data column.

A hint to create and use an index on the specified data column. The hint has default action.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function IndexedField(Of T)( _`
`  ByVal row As DataRow, _`
`  ByVal column As DataColumn _`
`) As T`

C#

`public static T IndexedField<T>(`
`  DataRow row,`
`  DataColumn column`
`)`

Parameters

row

The data row.

column

The data column.

Type Parameters

T

The type of the data column

Return Value

The data column value.

Remarks

Hints are used declaratively. They tell LiveLinq query optimizer to create and use an index on that column, if possible. When the query is executed, the hint method **IndexedField** is replaced with the standard LINQ to DataSet extension method Field. See C1.LiveLinq.Hints for more details.

Example

- C#

```
var query = from c in customersTable
    where c.IndexedField<string>(customerColumn) == "ALFKI"
    select c;
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

AdoNetExtensions Class
AdoNetExtensions Members
Overload List

IndexedField<T>(DataRow,String) Method
Example

The type of the data column

The data row.

The name of the column.

A hint to create and use an index on the specified data column. The hint has default action.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function IndexedField(Of T)( _
   ByVal row As DataRow, _
   ByVal columnName As String _
) As T
```

C#

```
public static T IndexedField<T>(
   DataRow row,
   string columnName
)
```

Parameters

row

The data row.

columnName

The name of the column.

Type Parameters

T

The type of the data column

Return Value

The data column value.

Remarks

Hints are used declaratively. They tell LiveLinq query optimizer to create and use an index on that column, if possible. When the query is executed, the hint method **IndexedField** is replaced with the standard LINQ to DataSet extension method Field. See C1.LiveLinq.Hints for more details.

Example

- C#

```
var query = from c in customersTable
```

```
    where c.IndexedField<string>("CustomerID") == "ALFKI"
    select c;
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

AdoNetExtensions Class
AdoNetExtensions Members
Overload List

IndexedField<T>(DataRow,Int32,IndexingHintAction) Method
Example

The type of the data column

The data row.

The zero-based ordinal position of the column.

The action specified by the hint.

A hint to create and use an index on the specified data column. The hint has specified action.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function IndexedField(Of T)( _
   ByVal row As DataRow, _
   ByVal columnIndex As Integer, _
   ByVal action As IndexingHintAction _
) As T
```

C#

```
public static T IndexedField<T>(
   DataRow row,
   int columnIndex,
```

```
    IndexingHintAction action
)
```

Parameters

row

The data row.

columnIndex

The zero-based ordinal position of the column.

action

The action specified by the hint.

Type Parameters

T

The type of the data column

Return Value

The data column value.

Remarks

Hints are used declaratively. They tell LiveLinq query optimizer to create and use an index on that column, if possible. When the query is executed, the hint method **IndexedField** is replaced with the standard LINQ to DataSet extension method Field. See C1.LiveLinq.Hints for more details.

Example

- C#

```
var query = from c in customersTable
    where c.IndexedField<string>(0,
IndexingHintAction.Mandatory) == "ALFKI"
    select c;
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

AdoNetExtensions Class
AdoNetExtensions Members
Overload List

IndexedField<T>(DataRow,DataColumn,IndexingHintAction) Method
Example

The type of the data column

The data row.

The data column.

The action specified by the hint.

A hint to create and use an index on the specified data column. The hint has specified action.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function IndexedField(Of T)( _`
`  ByVal row As DataRow, _`
`  ByVal column As DataColumn, _`
`  ByVal action As IndexingHintAction _`
`) As T`

C#

`public static T IndexedField<T>(`
`  DataRow row,`
`  DataColumn column,`
`  IndexingHintAction action`
`)`

Parameters

row

The data row.

column

The data column.

action

The action specified by the hint.

Type Parameters

T

The type of the data column

Return Value

The data column value.

Remarks

Hints are used declaratively. They tell LiveLinq query optimizer to create and use an index on that column, if possible. When the query is executed, the hint method **IndexedField** is replaced with the standard LINQ to DataSet extension method Field. See C1.LiveLinq.Hints for more details.

Example

- C#

```
var query =
    from c in customersTable
    where c.IndexedField<string>(customerColumn,
IndexingHintAction.Mandatory) == "ALFKI"
    select c;
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

AdoNetExtensions Class
AdoNetExtensions Members
Overload List

The type of the data column

The data row.

The name of the column.

The action specified by the hint.

A hint to create and use an index on the specified data column. The hint has specified action.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function IndexedField(Of T)( _`
`  ByVal row As DataRow, _`
`  ByVal columnName As String, _`
`  ByVal action As IndexingHintAction _`
`) As T`

C#

`public static T IndexedField<T>(`
`  DataRow row,`
`  string columnName,`
`  IndexingHintAction action`
`)`

Parameters

row

The data row.

columnName

The name of the column.

action

The action specified by the hint.

Type Parameters

T

The type of the data column

Return Value

The data column value.

Remarks

Hints are used declaratively. They tell LiveLinq query optimizer to create and use an index on that column, if possible. When the query is executed, the hint method **IndexedField** is replaced with the standard LINQ to DataSet extension method Field. See C1.LiveLinq.Hints for more details.

Example

- C#

```
var query =
    from c in customersTable
    where c.IndexedField<string>("CustomerID",
IndexingHintAction.Mandatory) == "ALFKI"
    select c;
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

AdoNetExtensions Class
AdoNetExtensions Members
Overload List

IndexedDataTable<TRow>
The type of the rows of the System.Data.DataTable. It is a class derived from System.Data.DataRow

A wrapper for the standard ADO.NET System.Data.DataTable class allowing to use ADO.NET data sources in LiveLinq indexing and live views.

Object Model

IndexedDataTable<TRow>

Syntax

Visual Basic (Declaration)

`Public Class IndexedDataTable(Of TRow As DataRow)`
`  Implements C1.LiveLinq.Indexing.IIndexedSource(Of TRow),`
`C1.LiveLinq.IObservableSource(Of TRow)`

C#

`public class IndexedDataTable<TRow> : C1.LiveLinq.Indexing.IIndexedSource<TRow>,`
`C1.LiveLinq.IObservableSource<TRow>`
`where TRow: DataRow`

Type Parameters

TRow

The type of the rows of the System.Data.DataTable. It is a class derived from System.Data.DataRow

Remarks

Note: The **IndexedDataTable** wrapper is owned by the original System.Data.DataTable object (in fact, it is stored in its **ExtendedProperties**). So, if you create a wrapper for the same data table several times, it will be the same object.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.AdoNet.IndexedDataTable<TRow>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedDataTable<TRow> Members
C1.LiveLinq.AdoNet Namespace

Overview

The type of the rows of the System.Data.DataTable. It is a class derived from System.Data.DataRow

A wrapper for the standard ADO.NET System.Data.DataTable class allowing to use ADO.NET data sources in LiveLinq indexing and live views.

Object Model

Syntax

Visual Basic (Declaration)

`Public Class IndexedDataTable(Of TRow As DataRow)`
`  Implements C1.LiveLinq.Indexing.IIndexedSource(Of TRow),`
`C1.LiveLinq.IObservableSource(Of TRow)`

C#

`public class IndexedDataTable<TRow> : C1.LiveLinq.Indexing.IIndexedSource<TRow>,`
`C1.LiveLinq.IObservableSource<TRow>`
`where TRow: DataRow`

Type Parameters

TRow

The type of the rows of the System.Data.DataTable. It is a class derived from System.Data.DataRow

Remarks

Note: The **IndexedDataTable** wrapper is owned by the original System.Data.DataTable object (in fact, it is stored in its **ExtendedProperties**). So, if you create a wrapper for the same data table several times, it will be the same object.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.AdoNet.IndexedDataTable<TRow>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedDataTable<TRow> Members
C1.LiveLinq.AdoNet Namespace

Members
Properties Methods

The following tables list the members exposed by IndexedDataTable<TRow>.

Public Properties

	Name	Description
	Count	Gets the number of rows in the data table.
	Indexes	The collection of indexes for this IndexedDataTable<TRow>
	Item	Gets the row at the specified ordinal position.
	Table	Gets the ADO.NET System.Data.DataTable object represented by this IndexedDataTable<TRow>.

Top

Public Methods

	Name	Description
	BeginUpdate	Suspends notifications while massive changes are being made to a data table.

	EndUpdate	Ends notification suspension started with BeginUpdate.
	GetEnumerator	Returns an enumerator that iterates through the IndexedDataTable<TRow>.

Top

See Also

Reference

IndexedDataTable<TRow> Class
C1.LiveLinq.AdoNet Namespace

Methods
For a list of all members of this type, see IndexedDataTable<TRow> members.

Public Methods

	Name	Description
	BeginUpdate	Suspends notifications while massive changes are being made to a data table.
	EndUpdate	Ends notification suspension started with BeginUpdate.
	GetEnumerator	Returns an enumerator that iterates through the IndexedDataTable<TRow>.

Top

See Also

Reference

IndexedDataTable<TRow> Class
C1.LiveLinq.AdoNet Namespace

BeginUpdate Method
Suspends notifications while massive changes are being made to a data table.

Syntax

Visual Basic (Declaration)
`Public Sub BeginUpdate()`
C#
`public void BeginUpdate()`

Remarks

This method must be followed by EndUpdate.

Use this method when you already have indexes built over a data table or live views based on that data table, and you need to re-populate it or perform other massive changes to rows of that data table. Without this method, every single change you make causes LiveLinq to perform necessary operations for maintaining your indexes and live views dependent on this data table. In case of massive changes, this can be slower than to wait until the massive changes are done and rebuild the indexes and live views.

Between **BeginUpdate** and EndUpdate calls, indexes, live views, bound controls and other change notification listeners are not updated, they don't receive change notifications.

When EndUpdate is called, a SourceChangeType.Modify or SourceChangeType.Reset notification is sent, depending on whether the change affected a single row or multiple rows of the data table. Even when you change a single row, it may make sense to enclose your changes in **BeginUpdate/EndUpdate** if you change multiple fields in the row. In that case a SourceChangeType.Modify notification is sent. If more than one row was changed, a SourceChangeType.Reset notification is sent, meaning all indexes, live views and other collections dependent on this data table must be rebuilt from scratch.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedDataTable<TRow> Class
IndexedDataTable<TRow> Members

EndUpdate Method

Ends notification suspension started with BeginUpdate.

Syntax

Visual Basic (Declaration)
`Public Sub EndUpdate()`
C#
`public void EndUpdate()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedDataTable<TRow> Class
IndexedDataTable<TRow> Members

GetEnumerator Method

Returns an enumerator that iterates through the IndexedDataTable<TRow>.

Syntax

Visual Basic (Declaration)
`Public Function GetEnumerator() As IEnumerator(Of TRow)`
C#
`public IEnumerator<TRow> GetEnumerator()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedDataTable<TRow> Class
IndexedDataTable<TRow> Members

Properties
For a list of all members of this type, see IndexedDataTable<TRow> members.

Public Properties

	Name	Description
	Count	Gets the number of rows in the data table.
	Indexes	The collection of indexes for this IndexedDataTable<TRow>
	Item	Gets the row at the specified ordinal position.
	Table	Gets the ADO.NET System.Data.DataTable object represented by this IndexedDataTable<TRow>.

Top

See Also

Reference

IndexedDataTable<TRow> Class
C1.LiveLinq.AdoNet Namespace

Count Property
Gets the number of rows in the data table.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Count As Integer`

C#

```
public int Count {get;}
```

Remarks

This property returns the same number of rows as the **Count** property of the System.Data.DataTable represented by this IndexedDataTable<TRow>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedDataTable<TRow> Class
IndexedDataTable<TRow> Members

Indexes Property
The collection of indexes for this IndexedDataTable<TRow>

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Property Indexes As IndexCollection(Of TRow)
```

C#

```
public IndexCollection<TRow> Indexes {get;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedDataTable<TRow> Class
IndexedDataTable<TRow> Members

Item Property
The zero-based ordinal position of the row to return.

Gets the row at the specified ordinal position.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Default Property Item( _
   ByVal ordinal As Integer _
) As TRow
```

C#

```
public TRow this[
   int ordinal
]; {get;}
```

Parameters

ordinal

The zero-based ordinal position of the row to return.

Property Value

The specified row.

Remarks

This property returns the same row as **Rows[ordinal]** of the System.Data.DataTable represented by this IndexedDataTable<TRow>.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedDataTable<TRow> Class
IndexedDataTable<TRow> Members

Table Property
Gets the ADO.NET System.Data.DataTable object represented by this IndexedDataTable<TRow>.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Table As DataTable`
C#
`public DataTable Table {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedDataTable<TRow> Class
IndexedDataTable<TRow> Members

C1.LiveLinq.Collections Namespace

Overview

Classes

	Class	Description
	IndexableObject	Base class for collection element classes.
	IndexedCollection<T>	Data collection class recommended for use in LiveLinq to Objects.

See Also

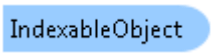
Reference

C1.LiveLinq.4 Assembly

Classes

IndexableObject
Base class for collection element classes.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class IndexableObject`
C#
`public class IndexableObject`

Remarks

Using IndexedCollection<T> and other collection classes|tag=LiveLinq to Objects: IndexedCollection(T) and other collection classes in LiveLinq to Objects, the element class **T** must implement the property change notification interface System.ComponentModel.INotifyPropertyChanged. The easiest way to satisfy this requirement is to derive that class from **IndexableObject**. Then you can use its method **OnPropertyChanged** to send the required property change notifications. For example, a **Customer** class can be defined like this: `public class Customer : IndexableObject { ..... private string _name; public string Name { get { return _name; } set { _name = value; OnPropertyChanged("Name"); } } ..... }`

Inheritance Hierarchy

System.Object
  C1.LiveLinq.Collections.IndexableObject
    C1.LiveLinq.LiveViews.ViewRow

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

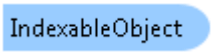
See Also

Reference

IndexableObject Members
C1.LiveLinq.Collections Namespace

Overview
Base class for collection element classes.

Object Model



Syntax

Visual Basic (Declaration)	
`Public Class IndexableObject`	
C#	
`public class IndexableObject`	

Remarks

Using IndexedCollection<T> and other collection classes|tag=LiveLinq to Objects: IndexedCollection(T) and other collection classes in LiveLinq to Objects, the element class **T** must implement the property change notification interface System.ComponentModel.INotifyPropertyChanged. The easiest way to satisfy this requirement is to derive that class from **IndexableObject**. Then you can use its method **OnPropertyChanged** to send the required property change notifications. For example, a **Customer** class can be defined like this: `public class Customer : IndexableObject { ..... private string _name; public string Name { get { return _name; } set { _name = value; OnPropertyChanged("Name"); } } ..... }`

Inheritance Hierarchy

System.Object
C1.LiveLinq.Collections.IndexableObject
C1.LiveLinq.LiveViews.ViewRow

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexableObject Members
C1.LiveLinq.Collections Namespace

Members
Events

The following tables list the members exposed by IndexableObject.

Public Constructors

	Name	Description
	IndexableObject Constructor	Initializes a new instance of the IndexableObject class.

Top

Public Events

	Name	Description
	PropertyChanged	Occurs when a property value changes, after it has been changed.

Top

See Also

Reference

IndexableObject Class
C1.LiveLinq.Collections Namespace

IndexableObject Constructor
Initializes a new instance of the IndexableObject class.

Syntax

Visual Basic (Declaration)
`Public Function New()`
C#
`public IndexableObject()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexableObject Class
IndexableObject Members

Events
For a list of all members of this type, see IndexableObject members.

Public Events

	Name	Description
	PropertyChanged	Occurs when a property value changes, after it has been changed.

Top

See Also

Reference

IndexableObject Class
C1.LiveLinq.Collections Namespace

PropertyChanged Event
Occurs when a property value changes, after it has been changed.

Syntax

Visual Basic (Declaration)	
`Public Event PropertyChanged As PropertyChangedEventHandler`	
C#	
`public event PropertyChangedEventHandler PropertyChanged`	

Event Data

The event handler receives an argument of type PropertyChangedEventArgs containing data related to this event. The following **PropertyChangedEventArgs** properties provide information specific to this event.

Property	Description
PropertyName	

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexableObject Class
IndexableObject Members

IndexedCollection<T>
The type of the elements in the collection.

Data collection class recommended for use in LiveLinq to Objects.

Object Model

IndexedCollection<T>

Syntax

Visual Basic (Declaration)	

`Public Class IndexedCollection(Of T)`
`  Inherits System.Collections.ObjectModel.Collection(Of T)`
`  Implements C1.LiveLinq.Indexing.IIndexedSource(Of T),`
`C1.LiveLinq.IObservableSource(Of T)`	

C#	

`public class IndexedCollection<T> :`
`System.Collections.ObjectModel.Collection<T>,`
`C1.LiveLinq.Indexing.IIndexedSource<T>, C1.LiveLinq.IObservableSource<T>`	

Type Parameters

T

The type of the elements in the collection.

Remarks

If you don't have preexisting collection classes and need to create your own, the best choice is to use the built-in LiveLinq collection class **IndexedCollection<T>.** It is specifically optimized for LiveLinq use.

Usually, the element class **T** must implement the property change notification interface System.ComponentModel.INotifyPropertyChanged. The easiest way to implement System.ComponentModel.INotifyPropertyChanged is to derive the element class from IndexableObject.

In rare cases where implementing System.ComponentModel.INotifyPropertyChanged is impossible, a custom C1.LiveLinq.Listeners.PropertyChangeListener<T> can be provided instead.

For details, see Using the built-in collection class IndexedCollection(T) (LiveLinq to Objects).

For alternative options in LiveLinq to Objects, see LiveLinq to Objects: IndexedCollection(T) and other collection classes

The **IndexedCollection<T>** class is not needed in LiveLinq to DataSet and LiveLinq to XML, because in those cases LiveLinq works with collections that already exist in ADO.NET and XML.

Inheritance Hierarchy

System.Object
  System.Collections.ObjectModel.Collection<T>
    C1.LiveLinq.Collections.IndexedCollection<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

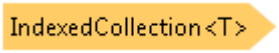
Reference

IndexedCollection<T> Members
C1.LiveLinq.Collections Namespace

Overview
The type of the elements in the collection.

Data collection class recommended for use in LiveLinq to Objects.

Object Model



Syntax

Visual Basic (Declaration)

`Public Class IndexedCollection(Of T)`
`  Inherits System.Collections.ObjectModel.Collection(Of T)`
`  Implements C1.LiveLinq.Indexing.IIndexedSource(Of T),`
`C1.LiveLinq.IObservableSource(Of T)`

C#
`public class IndexedCollection<T> :`

```
System.Collections.ObjectModel.Collection<T>,
C1.LiveLinq.Indexing.IIndexedSource<T>, C1.LiveLinq.IObservableSource<T>
```

Type Parameters

T

The type of the elements in the collection.

Remarks

If you don't have preexisting collection classes and need to create your own, the best choice is to use the built-in LiveLinq collection class **IndexedCollection<T>.** It is specifically optimized for LiveLinq use.

Usually, the element class **T** must implement the property change notification interface System.ComponentModel.INotifyPropertyChanged. The easiest way to implement System.ComponentModel.INotifyPropertyChanged is to derive the element class from IndexableObject.

In rare cases where implementing System.ComponentModel.INotifyPropertyChanged is impossible, a custom C1.LiveLinq.Listeners.PropertyChangeListener<T> can be provided instead.

For details, see Using the built-in collection class IndexedCollection(T) (LiveLinq to Objects).

For alternative options in LiveLinq to Objects, see LiveLinq to Objects: IndexedCollection(T) and other collection classes

The **IndexedCollection<T>** class is not needed in LiveLinq to DataSet and LiveLinq to XML, because in those cases LiveLinq works with collections that already exist in ADO.NET and XML.

Inheritance Hierarchy

System.Object
  System.Collections.ObjectModel.Collection<T>
    C1.LiveLinq.Collections.IndexedCollection<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedCollection<T> Members
C1.LiveLinq.Collections Namespace

Members
Properties Methods Events

The following tables list the members exposed by IndexedCollection<T>.

Public Constructors

	Name	Description
	IndexedCollection<T> Constructor	Overloaded.

Top

Public Properties

	Name	Description
	Count	(Inherited from System.Collections.ObjectModel.Collection<T>)
	CreateNew	Gets or sets a delegate that is used to create new items. If it is null, a public parameterless constructor of type **T** is used.
	Indexes	The collection of indexes for this IndexedCollection<T>
	Item	(Inherited from System.Collections.ObjectModel.Collection<T>)

Top

Public Methods

	Name	Description
	Add	(Inherited from System.Collections.ObjectModel.Collection<T>)
	AddRange	Adds the elements of the specified collection to the end of the

		IndexedCollection<T>
	BeginUpdate	Suspends notifications while massive changes are being made to the IndexedCollection<T>.
	Clear	(Inherited from System.Collections.ObjectModel.Collection<T>)
	Contains	(Inherited from System.Collections.ObjectModel.Collection<T>)
	CopyTo	(Inherited from System.Collections.ObjectModel.Collection<T>)
	EndUpdate	Ends notification suspension started with BeginUpdate.
	GetEnumerator	(Inherited from System.Collections.ObjectModel.Collection<T>)
	IndexOf	(Inherited from System.Collections.ObjectModel.Collection<T>)
	Insert	(Inherited from System.Collections.ObjectModel.Collection<T>)
	Remove	(Inherited from System.Collections.ObjectModel.Collection<T>)
	RemoveAt	(Inherited from System.Collections.ObjectModel.Collection<T>)

Top

Public Events

	Name	Description
	Changed	Occurs after the collection has changed.

Top

See Also

Reference

IndexedCollection<T> Class
C1.LiveLinq.Collections Namespace

IndexedCollection<T> Constructor

Overload List

Overload	Description
IndexedCollection<T> Constructor()	Initializes a new instance of the IndexedCollection<T> class.
IndexedCollection<T> Constructor(IList<T>)	Initializes a new instance of the IndexedCollection<T> class that contains elements copied from the specified list.
IndexedCollection<T> Constructor(IList<T>,PropertyChangeListener<T>)	Initializes a new instance of the IndexedCollection<T> class that contains elements copied from the specified list, and provides a custom C1.LiveLinq.Listeners.PropertyChangeListener<T>.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedCollection<T> Class
IndexedCollection<T> Members

IndexedCollection<T> Constructor()
Initializes a new instance of the IndexedCollection<T> class.

Syntax

Visual Basic (Declaration)

```
Public Function New()
```

C#

```
public IndexedCollection<T>()
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedCollection<T> Class
IndexedCollection<T> Members
Overload List

IndexedCollection<T> Constructor(IList<T>)
The collection whose elements are copied to the new instance.

Initializes a new instance of the IndexedCollection<T> class that contains elements copied from the specified list.

Syntax

Visual Basic (Declaration)

```
Public Function New( _
   ByVal list As IList(Of T) _
)
```

C#

```
public IndexedCollection<T>(
   IList<T> list
)
```

Parameters

list

The collection whose elements are copied to the new instance.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedCollection<T> Class
IndexedCollection<T> Members
Overload List

IndexedCollection<T> Constructor(IList<T>,PropertyChangeListener<T>)
The collection whose elements are copied to the new instance.

The custom C1.LiveLinq.Listeners.PropertyChangeListener<T> to use with the collection.

Initializes a new instance of the IndexedCollection<T> class that contains elements copied from the specified list, and provides a custom C1.LiveLinq.Listeners.PropertyChangeListener<T>.

Syntax

Visual Basic (Declaration)

```
Public Function New( _
   ByVal list As IList(Of T), _
   ByVal itemPropertyChangeListener As PropertyChangeListener(Of T) _
)
```

C#

```
public IndexedCollection<T>(
   IList<T> list,
   PropertyChangeListener<T> itemPropertyChangeListener
)
```

Parameters

list

The collection whose elements are copied to the new instance.

itemPropertyChangeListener

The custom C1.LiveLinq.Listeners.PropertyChangeListener<T> to use with the collection.

Remarks

This constructor is intended for the rare cases where implementing System.ComponentModel.INotifyPropertyChanged in the class of the elements of the collection is impossible

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedCollection<T> Class
IndexedCollection<T> Members
Overload List

Methods
For a list of all members of this type, see IndexedCollection<T> members.

Public Methods

	Name	Description
	Add	(Inherited from System.Collections.ObjectModel.Collection<T>)
	AddRange	Adds the elements of the specified collection to the end of the IndexedCollection<T>
	BeginUpdate	Suspends notifications while massive changes are being made to the IndexedCollection<T>.
	Clear	(Inherited from System.Collections.ObjectModel.Collection<T>)

	Contains	(Inherited from System.Collections.ObjectModel.Collection<T>)
	CopyTo	(Inherited from System.Collections.ObjectModel.Collection<T>)
	EndUpdate	Ends notification suspension started with BeginUpdate.
	GetEnumerator	(Inherited from System.Collections.ObjectModel.Collection<T>)
	IndexOf	(Inherited from System.Collections.ObjectModel.Collection<T>)
	Insert	(Inherited from System.Collections.ObjectModel.Collection<T>)
	Remove	(Inherited from System.Collections.ObjectModel.Collection<T>)
	RemoveAt	(Inherited from System.Collections.ObjectModel.Collection<T>)

Top

See Also

Reference

IndexedCollection<T> Class
C1.LiveLinq.Collections Namespace

AddRange Method
The collection whose elements should be added to the end of the IndexedCollection<T>

Adds the elements of the specified collection to the end of the IndexedCollection<T>

Syntax

Visual Basic (Declaration)

```
Public Sub AddRange( _
   ByVal items As IEnumerable(Of T) _
)
```

C#

```
public void AddRange(
   IEnumerable<T> items
)
```

Parameters

items

The collection whose elements should be added to the end of the IndexedCollection<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedCollection<T> Class
IndexedCollection<T> Members

BeginUpdate Method

Suspends notifications while massive changes are being made to the IndexedCollection<T>.

Syntax

Visual Basic (Declaration)
`Public Sub BeginUpdate()`
C#
`public void BeginUpdate()`

Remarks

This method must be followed by EndUpdate.

Use this method when you already have indexes built over the IndexedCollection<T> or live views based on it, and you need to re-populate this IndexedCollection<T> or perform other massive changes to items of this collection. Without this method, every single change you make causes LiveLinq to perform necessary operations for maintaining your indexes and live views dependent on

this collection. In case of massive changes, this can be slower than to wait until the massive changes are done and rebuild the indexes and live views.

Between **BeginUpdate** and EndUpdate calls, indexes, live views, bound controls and other change notification listeners are not updated, they don't receive change notifications.

When EndUpdate is called, a SourceChangeType.Modify or SourceChangeType.Reset notification is sent, depending on whether the change affected a single item or multiple items of the collection. Even when you change a single item, it may make sense to enclose your changes in **BeginUpdate/EndUpdate** if you change multiple properties of the item. In that case a SourceChangeType.Modify notification is sent. If more than one item was changed, a SourceChangeType.Reset notification is sent, meaning all indexes, live views and other collections dependent on this data table must be rebuilt from scratch.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedCollection<T> Class
IndexedCollection<T> Members

EndUpdate Method
Ends notification suspension started with BeginUpdate.

Syntax

Visual Basic (Declaration)
`Public Sub EndUpdate()`
C#
`public void EndUpdate()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedCollection<T> Class
IndexedCollection<T> Members

Properties
For a list of all members of this type, see IndexedCollection<T> members.

Public Properties

	Name	Description
	Count	(Inherited from System.Collections.ObjectModel.Collection<T>)
	CreateNew	Gets or sets a delegate that is used to create new items. If it is null, a public parameterless constructor of type **T** is used.
	Indexes	The collection of indexes for this IndexedCollection<T>
	Item	(Inherited from System.Collections.ObjectModel.Collection<T>)

Top

See Also

Reference

IndexedCollection<T> Class
C1.LiveLinq.Collections Namespace

CreateNew Property
Gets or sets a delegate that is used to create new items. If it is null, a public parameterless constructor of type **T** is used.

Syntax

Visual Basic (Declaration)
`Public Property CreateNew As Func(Of T)`

C#

```
public Func<T> CreateNew {get; set;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedCollection<T> Class
IndexedCollection<T> Members

Indexes Property
The collection of indexes for this IndexedCollection<T>

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Property Indexes As IndexCollection(Of T)
```

C#

```
public IndexCollection<T> Indexes {get;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedCollection<T> Class
IndexedCollection<T> Members

Events
For a list of all members of this type, see IndexedCollection<T> members.

Public Events

	Name	Description
	Changed	Occurs after the collection has changed.

Top

See Also

Reference

IndexedCollection<T> Class
C1.LiveLinq.Collections Namespace

Changed Event
Occurs after the collection has changed.

Syntax

Visual Basic (Declaration)
`Public Event Changed As EventHandler(Of SourceChangeEventArgs(Of T))`
C#
`public event EventHandler<SourceChangeEventArgs<T>> Changed`

Event Data

The event handler receives an argument of type SourceChangeEventArgs<T> containing data related to this event. The following **SourceChangeEventArgs<T>** properties provide information specific to this event.

Property	Description
ChangeType	Gets the type of change.
Item	Gets the object that is being changed.
Ordinal	Gets the ordinal position of the collection item that is being changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedCollection<T> Class
IndexedCollection<T> Members

C1.LiveLinq.Indexing Namespace

Overview

Classes

	Class	Description
	Index<T>	Base class for the Index<T,TKey> class.
	Index<T,TKey>	Indexes a collection by an expression (typically, by a field), providing fast access to items having particular values (or range of values) of that expression.
	IndexCollection<T>	Represents a collection of indexes attached to an indexed collection.
	IndexDefinition<T>	Contains common part of the Index and Subindex classes.
	IndexingAlgorithm	Defines the kind of an index, the algorithm used by that index. Currently, the RedBlackTree algorithm is always used.
	IndexingException	Represents an exception that is thrown when errors are generated using LiveLinq components.
	ScannerCollection<T>	Represents a collection of indexes or subindexes.

	Subindex<T>	Base class for the Subindex<T,TKey> class.
	Subindex<T,TKey>	Defines a subindex, an index definition subordinate to another index definition, its parent.
	SubindexCollection<T>	Represents a collection of subindexes attached to an IndexDefinition<T>.

Interfaces

	Interface	Description
	IIndexedSource<T>	Represents an indexed collection.

See Also

Reference

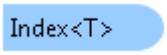
C1.LiveLinq.4 Assembly

Classes

Index<T>
The type of the elements of the collection to index.

Base class for the Index<T,TKey> class.

Object Model



Syntax

Visual Basic (Declaration)	

`Public MustInherit Class Index(Of T)`
`  Inherits IndexDefinition(Of T)`
`  Implements C1.LiveLinq.Indexing.Search.IIndexScanner(Of T)`	

C#	

```
public abstract class Index<T> : IndexDefinition<T>,
C1.LiveLinq.Indexing.Search.IIndexScanner<T>
```

Type Parameters

T

The type of the elements of the collection to index.

Remarks

You don't typically use the **Index<T>** class directly. It provides functionality of the Index<T,TKey> class that does not depend on the index key type. The base class **Index<T>** is needed only if the index key type is not known, usually in general-purpose code intended for reuse with different key types.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.Indexing.IndexDefinition<T>
    C1.LiveLinq.Indexing.Index<T>
      C1.LiveLinq.Indexing.Index<T,TKey>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T> Members
C1.LiveLinq.Indexing Namespace
Index<T,TKey> Class

Overview
The type of the elements of the collection to index.

Base class for the Index<T,TKey> class.

Object Model

Index<T>

Syntax

Visual Basic (Declaration)	

`Public MustInherit Class Index(Of T)`
`  Inherits IndexDefinition(Of T)`
`  Implements C1.LiveLinq.Indexing.Search.IIndexScanner(Of T)`	

C#	
`public abstract class Index<T> : IndexDefinition<T>,` `C1.LiveLinq.Indexing.Search.IIndexScanner<T>`	

Type Parameters

T

The type of the elements of the collection to index.

Remarks

You don't typically use the **Index<T>** class directly. It provides functionality of the Index<T,TKey> class that does not depend on the index key type. The base class **Index<T>** is needed only if the index key type is not known, usually in general-purpose code intended for reuse with different key types.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.Indexing.IndexDefinition<T>
    C1.LiveLinq.Indexing.Index<T>
      C1.LiveLinq.Indexing.Index<T,TKey>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T> Members
C1.LiveLinq.Indexing Namespace
Index<T,TKey> Class

Members
Properties Methods

The following tables list the members exposed by Index<T>.

Public Properties

	Name	Description
	Algorithm	Gets the indexing algorithm used by the index. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	ItemCount	Gets the number of elements in the indexed collection.
	KeyCount	Gets the number of distinct key values in all items of this collection.
	KeyIsUnique	Gets a value that indicates whether the key used in this index is a unique key for the collection. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	KeySelector	Gets the expression used to obtain key value from an element of the indexed collection. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	KeyType	Gets the type of the index key. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	Locale	Gets the locale information used to compare strings in the index. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	Root	Gets the root index in an index/subindex hierarchy. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	Subindexes	Gets the collection of subindexes added to this index. (Inherited from

		C1.LiveLinq.Indexing.IndexDefinition<T>)

Top

Public Methods

	Name	Description
	ContainsKey	Returns a value that indicates whether the indexed collection contains an item with the given key value. Implements IIndexScanner(T).ContainsKey(object)
	Find	Finds items with the specified key value. Implements IIndexScanner(T).Find(object)
	FindBetween	Finds items with key values in the interval between the specified values. Implements IIndexScanner(T).FindBetween(object,bool,object,bool,Order)
	FindGreater	Finds items with keys greater than the specified value. Implements IIndexScanner(T).FindGreater(object,bool,Order)
	FindKeys	Finds items containing any of the specified key values. Implements FindKeys(IEnumerable,Order)
	FindLess	Finds items with keys less than the specified value.

		Implements IIndexScanner(T).FindLess(object,bool,Order)
	FindStartingWith	Finds items with string key values starting with the specified string. Implements IIndexScanner(T).FindStartingWith(string,Func(string, bool),Order)
	GroupJoin	Overloaded. Correlates the items of this indexed collection with the items of another indexed collection and groups the results by the item of this collection. Implements IIndexScanner(T).GroupJoin
	Join	Overloaded. Correlates the items of this indexed collection with the items of another sequence and returns the combined items with matching keys. Implements IIndexScanner(T).Join

Top

See Also

Reference

Index<T> Class
C1.LiveLinq.Indexing Namespace
Index<T,TKey> Class

Methods
For a list of all members of this type, see Index<T> members.

Public Methods

	Name	Description
	ContainsKey	Returns a value that indicates whether the indexed collection contains an item with the given key value. Implements IIndexScanner(T).ContainsKey(object)
	Find	Finds items with the specified key value. Implements IIndexScanner(T).Find(object)
	FindBetween	Finds items with key values in the interval between the specified values. Implements IIndexScanner(T).FindBetween(object,bool,object,bool,Order)
	FindGreater	Finds items with keys greater than the specified value. Implements IIndexScanner(T).FindGreater(object,bool,Order)
	FindKeys	Finds items containing any of the specified key values. Implements FindKeys(IEnumerable,Order)
	FindLess	Finds items with keys less than the specified value. Implements IIndexScanner(T).FindLess(object,bool,Order)

	FindStartingWith	Finds items with string key values starting with the specified string. Implements IIndexScanner(T).FindStartingWith(string,Func(string, bool),Order)
	GroupJoin	Overloaded. Correlates the items of this indexed collection with the items of another indexed collection and groups the results by the item of this collection. Implements IIndexScanner(T).GroupJoin
	Join	Overloaded. Correlates the items of this indexed collection with the items of another sequence and returns the combined items with matching keys. Implements IIndexScanner(T).Join

Top

See Also

Reference

Index<T> Class
C1.LiveLinq.Indexing Namespace
Index<T,TKey> Class

ContainsKey Method
The key value to search for.

Returns a value that indicates whether the indexed collection contains an item with the given key value.

Implements IIndexScanner(T).ContainsKey(object)

Syntax

Visual Basic (Declaration)

```
Public Function ContainsKey( _
   ByVal key As Object _
) As Boolean
```

C#

```
public bool ContainsKey(
   object key
)
```

Parameters

key

The key value to search for.

Return Value

true if the indexed collection contains an element with the specified key value; otherwise, **false**.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T> Class
Index<T> Members

Find Method
The key value to search for.

Finds items with the specified key value.

Implements IIndexScanner(T).Find(object)

Syntax

Visual Basic (Declaration)

```
Public Function Find( _
   ByVal key As Object _
) As IndexQuery(Of T)
```

C#

```
public IndexQuery<T> Find(
   object key
)
```

Parameters

key

The key value to search for.

Return Value

An object enumerating items having the specified key value.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T> Class
Index<T> Members

FindBetween Method
Minimum key value to search for.

If **true**, the result includes items with the minimum key value.

Maximum key value to search for.

If **true**, the result includes items with the maximum key value.

Optionally specifies the order of the key values to sort the result (C1.LiveLinq.Order.**Unordered** if sorting is not required).

Finds items with key values in the interval between the specified values.

Implements IIndexScanner(T).FindBetween(object,bool,object,bool,Order)

Syntax

Visual Basic (Declaration)

```
Public Function FindBetween( _
   ByVal min As Object, _
   ByVal minInclusive As Boolean, _
   ByVal max As Object, _
   ByVal maxInclusive As Boolean, _
   ByVal order As Order _
) As IndexQuery(Of T)
```

C#

```
public IndexQuery<T> FindBetween(
   object min,
   bool minInclusive,
   object max,
   bool maxInclusive,
   Order order
)
```

Parameters

min

Minimum key value to search for.

minInclusive

If **true**, the result includes items with the minimum key value.

max

Maximum key value to search for.

maxInclusive

If **true**, the result includes items with the maximum key value.

order

Optionally specifies the order of the key values to sort the result (C1.LiveLinq.Order.**Unordered** if sorting is not required).

Return Value

An object enumerating all items with key values within the specified limits.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T> Class
Index<T> Members

FindGreater Method

Minimum key value to search for.

If **true**, the result includes items with the specified key value. Otherwise, the result only includes those with keys strictly greater than the specified value.

Optionally specifies the order of the key values to sort the result (C1.LiveLinq.Order.**Unordered** if sorting is not required).

Finds items with keys greater than the specified value.

Implements IIndexScanner(T).FindGreater(object,bool,Order)

Syntax

Visual Basic (Declaration)

```
Public Function FindGreater( _
   ByVal key As Object, _
   ByVal inclusive As Boolean, _
   ByVal order As Order _
```

```
) As IndexQuery(Of T)
```

C#

```
public IndexQuery<T> FindGreater(
   object key,
   bool inclusive,
   Order order
)
```

Parameters

key

Minimum key value to search for.

inclusive

If **true**, the result includes items with the specified key value. Otherwise, the result only includes those with keys strictly greater than the specified value.

order

Optionally specifies the order of the key values to sort the result (C1.LiveLinq.Order.**Unordered** if sorting is not required).

Return Value

An object enumerating all items whose key values are greater than the specified value.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T> Class
Index<T> Members

FindKeys Method
The key values to search for.

Optionally specifies the order of the key values to sort the result (C1.LiveLinq.Order.**Unordered** if sorting is not required).

Finds items containing any of the specified key values.

Implements FindKeys(IEnumerable,Order)

Syntax

Visual Basic (Declaration)

```
Public Function FindKeys( _
   ByVal keys As IEnumerable, _
   ByVal order As Order _
) As IndexQuery(Of T)
```

C#

```
public IndexQuery<T> FindKeys(
   IEnumerable keys,
   Order order
)
```

Parameters

keys

The key values to search for.

order

Optionally specifies the order of the key values to sort the result (C1.LiveLinq.Order.**Unordered** if sorting is not required).

Return Value

An object enumerating all items whose key values belong to the specified key value collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T> Class
Index<T> Members

FindLess Method
Maximum key value to search for.

If **true**, the result includes items with the specified key value. Otherwise, the result only includes those with keys strictly less than the specified value.

Optionally specifies the order of the key values to sort the result (C1.LiveLinq.Order.**Unordered** if sorting is not required).

Finds items with keys less than the specified value.

Implements IIndexScanner(T).FindLess(object,bool,Order)

Syntax

Visual Basic (Declaration)

```
Public Function FindLess( _
   ByVal key As Object, _
   ByVal inclusive As Boolean, _
   ByVal order As Order _
) As IndexQuery(Of T)
```

C#

```
public IndexQuery<T> FindLess(
   object key,
   bool inclusive,
   Order order
)
```

Parameters

key

Maximum key value to search for.

inclusive

If **true**, the result includes items with the specified key value. Otherwise, the result only includes those with keys strictly less than the specified value.

order

Optionally specifies the order of the key values to sort the result (C1.LiveLinq.Order.**Unordered** if sorting is not required).

Return Value

An object enumerating all items whose key values are less than the specified value.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T> Class
Index<T> Members

FindStartingWith Method

The string to search for as the beginning of key value strings.

An optional condition that found items must satisfy.

Optionally specifies the order of the key values to sort the result (C1.LiveLinq.Order.**Unordered** if sorting is not required).

Finds items with string key values starting with the specified string.

Implements IIndexScanner(T).FindStartingWith(string,Func(string, bool),Order)

Syntax

Visual Basic (Declaration)

```
Public Function FindStartingWith( _
   ByVal value As String, _
   ByVal keyPredicate As Func(Of String,Boolean), _
   ByVal order As Order _
```

```
) As IndexQuery(Of T,String)
```

C#

```
public IndexQuery<T,string> FindStartingWith(
   string value,
   Func<string,bool> keyPredicate,
   Order order
)
```

Parameters

value

The string to search for as the beginning of key value strings.

keyPredicate

An optional condition that found items must satisfy.

order

Optionally specifies the order of the key values to sort the result (C1.LiveLinq.Order.**Unordered** if sorting is not required).

Return Value

An object enumerating all items whose key values are strings that have a beginning matching the specified string and satisfy the optional condition.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T> Class
Index<T> Members

GroupJoin Method

Correlates the items of this indexed collection with the items of another indexed collection and groups the results by the item of this collection.

Implements IIndexScanner(T).GroupJoin

Overload List

Overload	Description
GroupJoin<T2,TResult>(IIndexScanner<T2>,Func<T,IEnumerable<T2>,TResult>)	Correlates the items of this indexed collection with the items of another indexed collection and groups the results by the item of this collection. Implements IIndexScanner(T).GroupJoin
GroupJoin<T2,TResult>(IEnumerable<T2>,Func<T2,Object>,Func<IEnumerable<T>,T2,TResult>)	Correlates the items of this indexed collection with the items of another sequence and groups the results by the item of the second sequence. Implements IIndexScanner(T).GroupJoin
GroupJoin<T2,TResult>(IEnumerable<T2>,Func<T2,Object>,Func<T,IEnume	Correlates the items

rable<T2>,TResult>)	of this indexed collection with the items of another sequence and groups the results by the item of this collection. Implements IIndexScanner(T).GroupJoin

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T> Class
Index<T> Members

GroupJoin<T2,TResult>(IIndexScanner<T2>,Func<T,IEnumerable<T2>,TResult>) Method
The type of the elements of the second collection.

The type of the result elements.

The second indexed collection to join to this collection.

A function to create a result element from an element from this collection and a collection of matching elements from the second collection.

Correlates the items of this indexed collection with the items of another indexed collection and groups the results by the item of this collection.

Implements IIndexScanner(T).GroupJoin

Syntax

Visual Basic (Declaration)

```
Public Overloads Function GroupJoin
    (Of T2,TResult)( _
   ByVal source As IIndexScanner(Of T2), _
   ByVal resultSelector As Func(Of T,IEnumerable(Of T2),TResult) _
) As IEnumerable(Of TResult)
```

C#

```
public IEnumerable<TResult> GroupJoin<T2,TResult>(
   IIndexScanner<T2> source,
   Func<T,IEnumerable<T2>,TResult> resultSelector
)
```

Parameters

source

The second indexed collection to join to this collection.

resultSelector

A function to create a result element from an element from this collection and a collection of matching elements from the second collection.

Type Parameters

T2

The type of the elements of the second collection.

TResult

The type of the result elements.

Return Value

Enumeration of objects obtained by applying the result selector to group pairs, where each pair consists of an item of this collection and the corresponding enumeration of the items of the second collection joined to it.

Remarks

Matching of two elements is performed by matching their keys.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T> Class
Index<T> Members
Overload List

GroupJoin<T2,TResult>(IEnumerable<T2>,Func<T2,Object>,Func<IEnumerable<T>,T2,TResult>) Method
The type of the elements of the second sequence.

The type of the result elements.

The second sequence to join to this collection.

A function to extract from an item of the second sequence the value to match against this collection's key value.

A function to create a result element from an element of the second sequence and the collection of matching elements from this collection.

Correlates the items of this indexed collection with the items of another sequence and groups the results by the item of the second sequence.

Implements IIndexScanner(T).GroupJoin

Syntax

Visual Basic (Declaration)

```
Public Overloads Function GroupJoin
    (Of T2,TResult)( _
   ByVal source As IEnumerable(Of T2), _
   ByVal keySelector As Func(Of T2,Object), _
   ByVal resultSelector As Func(Of IEnumerable(Of T),T2,TResult) _
) As IEnumerable(Of TResult)
```

C#

```
public IEnumerable<TResult> GroupJoin<T2,TResult>(
   IEnumerable<T2> source,
   Func<T2,object> keySelector,
   Func<IEnumerable<T>,T2,TResult> resultSelector
)
```

Parameters

source

The second sequence to join to this collection.

keySelector

A function to extract from an item of the second sequence the value to match against this collection's key value.

resultSelector

A function to create a result element from an element of the second sequence and the collection of matching elements from this collection.

Type Parameters

T2

The type of the elements of the second sequence.

TResult

The type of the result elements.

Return Value

Enumeration of objects obtained by applying the result selector to group pairs, where each pair consists of an item of the second collection and the corresponding enumeration of the items of this collection joined to it.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T> Class
Index<T> Members
Overload List

GroupJoin<T2,TResult>(IEnumerable<T2>,Func<T2,Object>,Func<T,IEnumerable<T2>,TResult>) Method
The type of the elements of the second sequence.

The type of the result elements.

The second sequence to join to this collection.

A function to extract from an item of the second sequence the value to match against this collection's key value.

A function to create a result element from an element from this collection and a collection of matching elements from the second sequence.

Correlates the items of this indexed collection with the items of another sequence and groups the results by the item of this collection.

Implements IIndexScanner(T).GroupJoin

Syntax

Visual Basic (Declaration)

```
Public Overloads Function GroupJoin
    (Of T2,TResult)( _
   ByVal source As IEnumerable(Of T2), _
   ByVal keySelector As Func(Of T2,Object), _
   ByVal resultSelector As Func(Of T,IEnumerable(Of T2),TResult) _
) As IEnumerable(Of TResult)
```

C#

```
public IEnumerable<TResult> GroupJoin<T2,TResult>(
   IEnumerable<T2> source,
   Func<T2,object> keySelector,
   Func<T,IEnumerable<T2>,TResult> resultSelector
)
```

Parameters

source

The second sequence to join to this collection.

keySelector

A function to extract from an item of the second sequence the value to match against this collection's key value.

resultSelector

A function to create a result element from an element from this collection and a collection of matching elements from the second sequence.

Type Parameters

T2

The type of the elements of the second sequence.

TResult

The type of the result elements.

Return Value

Enumeration of objects obtained by applying the result selector to group pairs, where each pair consists of an item of this collection and the corresponding enumeration of the items of the second sequence joined to it.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T> Class
Index<T> Members
Overload List

Join Method
Correlates the items of this indexed collection with the items of another sequence and returns the combined items with matching keys.

Implements IIndexScanner(T).Join

Overload List

Overload	Description
Join<T2,TResult>(IEnumerable<T2>,Func<T2,Object>,Func<T,T2,TResult>,JoinOperator)	Correlates the items of this indexed collection with the items of another sequence and returns the combined items with matching keys. Implements IIndexScanner(T).Join
Join<T2,TResult>(IIndexScanner<T2>,Func<T,T2,TResult>,JoinOperator)	Correlates the items of this indexed collection with the items of another indexed collection and returns the combined items with matching keys. Implements IIndexScanner(T).Join

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T> Class
Index<T> Members

Join<T2,TResult>(IEnumerable<T2>,Func<T2,Object>,Func<T,T2,TResult>,JoinOperator) Method
The type of the elements of the second sequence.

The type of the result elements.

The second sequence to join to this collection.

A function to extract from a second sequence's item the value to match against this collection's key value.

A function to create a result element from two matching elements.

A comparison operator to match elements.

Correlates the items of this indexed collection with the items of another sequence and returns the combined items with matching keys.

Implements IIndexScanner(T).Join

Syntax

Visual Basic (Declaration)	

```
Public Overloads Function Join
    (Of T2,TResult)( _
   ByVal source As IEnumerable(Of T2), _
   ByVal keySelector As Func(Of T2,Object), _
   ByVal resultSelector As Func(Of T,T2,TResult), _
   ByVal op As JoinOperator _
```

```
) As IEnumerable(Of TResult)
```

C#

```
public IEnumerable<TResult> Join<T2,TResult>(
   IEnumerable<T2> source,
   Func<T2,object> keySelector,
   Func<T,T2,TResult> resultSelector,
   JoinOperator op
)
```

Parameters

source

The second sequence to join to this collection.

keySelector

A function to extract from a second sequence's item the value to match against this collection's key value.

resultSelector

A function to create a result element from two matching elements.

op

A comparison operator to match elements.

Type Parameters

T2

The type of the elements of the second sequence.

TResult

The type of the result elements.

Return Value

Enumeration of objects obtained by applying the result selector to pairs of joined elements of the two collections.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T> Class
Index<T> Members
Overload List

Join<T2,TResult>(IIndexScanner<T2>,Func<T,T2,TResult>,JoinOperator) Method
The type of the elements of the second collection.

The type of the result elements.

The second indexed collection to join to this collection.

A function to create a result element from two matching elements.

A comparison operator to match elements.

Correlates the items of this indexed collection with the items of another indexed collection and returns the combined items with matching keys.

Implements IIndexScanner(T).Join

Syntax

Visual Basic (Declaration)

```
Public Overloads Function Join
    (Of T2,TResult)( _
   ByVal source As IIndexScanner(Of T2), _
   ByVal resultSelector As Func(Of T,T2,TResult), _
   ByVal op As JoinOperator _
) As IEnumerable(Of TResult)
```

C#

```
public IEnumerable<TResult> Join<T2,TResult>(
   IIndexScanner<T2> source,
```

```
    Func<T,T2,TResult> resultSelector,
    JoinOperator op
)
```

Parameters

source

The second indexed collection to join to this collection.

resultSelector

A function to create a result element from two matching elements.

op

A comparison operator to match elements.

Type Parameters

T2

The type of the elements of the second collection.

TResult

The type of the result elements.

Return Value

Enumeration of objects obtained by applying the result selector to pairs of joined elements of the two collections.

Remarks

Matching of two elements is performed by matching their keys.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T> Class
Index<T> Members
Overload List

Properties
For a list of all members of this type, see Index<T> members.

Public Properties

	Name	Description
	Algorithm	Gets the indexing algorithm used by the index. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	ItemCount	Gets the number of elements in the indexed collection.
	KeyCount	Gets the number of distinct key values in all items of this collection.
	KeyIsUnique	Gets a value that indicates whether the key used in this index is a unique key for the collection. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	KeySelector	Gets the expression used to obtain key value from an element of the indexed collection. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	KeyType	Gets the type of the index key. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	Locale	Gets the locale information used to compare strings in the index. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	Root	Gets the root index in an index/subindex hierarchy. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	Subindexes	Gets the collection of subindexes added to this index. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)

Top

See Also

Reference

Index<T> Class
C1.LiveLinq.Indexing Namespace
Index<T,TKey> Class

ItemCount Property
Gets the number of elements in the indexed collection.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property ItemCount As Integer`
C#
`public int ItemCount {get;}`

Property Value

The number of elements in the indexed collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T> Class
Index<T> Members

KeyCount Property
Gets the number of distinct key values in all items of this collection.

Syntax

Visual Basic (Declaration)

```
Public MustOverride ReadOnly Property KeyCount As Integer
```

C#

```
public abstract int KeyCount {get;}
```

Property Value

Number of distinct key values in the collection.

Remarks

This number is not the same as ItemCount, unless the index key is a unique key of that collection, see IndexDefinition<T>.KeyIsUnique.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

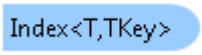
Index<T> Class
Index<T> Members

Index<T,TKey>
The type of the elements of the collection to index.

The type of the index key.

Indexes a collection by an expression (typically, by a field), providing fast access to items having particular values (or range of values) of that expression.

Object Model



Syntax

Visual Basic (Declaration)

```
Public MustInherit Class Index
    (Of T,TKey)
   Inherits Index(Of T)
   Implements C1.LiveLinq.Indexing.Search.IIndexScanner(Of T),
C1.LiveLinq.Indexing.Search.IIndexScanner(Of T,TKey)
```

C#

```
public abstract class Index<T,TKey> : Index<T>,
C1.LiveLinq.Indexing.Search.IIndexScanner<T>,
C1.LiveLinq.Indexing.Search.IIndexScanner<T,TKey>
```

Type Parameters

T

The type of the elements of the collection to index.

TKey

The type of the index key.

Remarks

Indexes can be created and added to a collection explicitly in code by calling IndexCollection.Add, or their creation can be enforced in LINQ queries by using the Indexed hint.

In LINQ queries, indexes are used for optimizing query performance if that is specified with an Indexed hint. Usually, hints are not required, because LiveLinq can automatically determine that an index can be used to speedup a query, but using hints helps to ensure this optimization.

Indexes can also be used programmatically in code, without LINQ syntax, by using the methods of the C1.LiveLinq.Indexing.Search.IIndexScanner<T,TKey> interface that is implemented by the Index<T,TKey> class. For example, you can call such methods as Find directly for an Index<T,TKey> object.

It must be kept in mind that every index you create introduces a trade-off: it can dramatically speed up searches, but it consumes memory and adds some (usually, small) overhead every time the indexed collection (or any of the items, its elements) is modified. This is why indexes should generally be kept only while you need them for queries. To delete an index, use IndexCollection.Remove.

An index can have subindexes, see Subindex<T,TKey>. Subindexes are optional, not required for any indexing tasks, but can provide additional optimization and help minimize memory requirements when a collection is indexed by multiple keys. In presence of subindexes, an index is the root level of a tree of subindexes.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.Indexing.IndexDefinition<T>
    C1.LiveLinq.Indexing.Index<T>
      C1.LiveLinq.Indexing.Index<T,TKey>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T,TKey> Members
C1.LiveLinq.Indexing Namespace

Overview
The type of the elements of the collection to index.

The type of the index key.

Indexes a collection by an expression (typically, by a field), providing fast access to items having particular values (or range of values) of that expression.

Object Model

Syntax

Visual Basic (Declaration)	
`Public MustInherit Class Index` `    (Of T,TKey)`	

```
   Inherits Index(Of T)
   Implements C1.LiveLinq.Indexing.Search.IIndexScanner(Of T),
C1.LiveLinq.Indexing.Search.IIndexScanner(Of T,TKey)
```

C#

```
public abstract class Index<T,TKey> : Index<T>,
C1.LiveLinq.Indexing.Search.IIndexScanner<T>,
C1.LiveLinq.Indexing.Search.IIndexScanner<T,TKey>
```

Type Parameters

T

The type of the elements of the collection to index.

TKey

The type of the index key.

Remarks

Indexes can be created and added to a collection explicitly in code by calling IndexCollection.Add, or their creation can be enforced in LINQ queries by using the Indexed hint.

In LINQ queries, indexes are used for optimizing query performance if that is specified with an Indexed hint. Usually, hints are not required, because LiveLinq can automatically determine that an index can be used to speedup a query, but using hints helps to ensure this optimization.

Indexes can also be used programmatically in code, without LINQ syntax, by using the methods of the C1.LiveLinq.Indexing.Search.IIndexScanner<T,TKey> interface that is implemented by the Index<T,TKey> class. For example, you can call such methods as Find directly for an Index<T,TKey> object.

It must be kept in mind that every index you create introduces a trade-off: it can dramatically speed up searches, but it consumes memory and adds some (usually, small) overhead every time the indexed collection (or any of the items, its elements) is modified. This is why indexes should generally be kept only while you need them for queries. To delete an index, use IndexCollection.Remove.

An index can have subindexes, see Subindex<T,TKey>. Subindexes are optional, not required for any indexing tasks, but can provide additional optimization and help

minimize memory requirements when a collection is indexed by multiple keys. In presence of subindexes, an index is the root level of a tree of subindexes.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.Indexing.IndexDefinition<T>
    C1.LiveLinq.Indexing.Index<T>
      C1.LiveLinq.Indexing.Index<T,TKey>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T,TKey> Members
C1.LiveLinq.Indexing Namespace

Members
Properties Methods

The following tables list the members exposed by Index<T,TKey>.

Public Properties

	Name	Description
	Algorithm	Gets the indexing algorithm used by the index. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	ItemCount	Gets the number of elements in the indexed collection. (Inherited from C1.LiveLinq.Indexing.Index<T>)
	KeyCount	Gets the number of distinct key values in all items of this collection. (Inherited from C1.LiveLinq.Indexing.Index<T>)

	KeyIsUnique	Gets a value that indicates whether the key used in this index is a unique key for the collection. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	KeySelector	Gets the expression used to obtain key value from an element of the indexed collection.
	KeyType	Gets the type of the index key. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	Locale	Gets the locale information used to compare strings in the index. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	Root	Gets the root index in an index/subindex hierarchy. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	Subindexes	Gets the collection of subindexes added to this index. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)

Top

Public Methods

	Name	Description
	All	Gets all items in the indexed collection.
	ContainsKey	Overloaded. Returns a value that indicates whether the collection contains an item with the given key value.
	Find	Overloaded. Finds items with the specified key value.
	FindBetween	Overloaded. Finds items with key values in the interval between the specified values.
	FindGreater	Overloaded. Finds items with keys greater than the specified value.

	FindKeys	Overloaded. Finds items containing any of the specified key values.
	FindLess	Overloaded. Finds items with keys less than the specified value.
	FindSingle	Finds the only item with the specified key value and throws an exception if there is not exactly one item with that key value.
	FindStartingWith	Finds items with string key values starting with the specified string. Implements IIndexScanner(T).FindStartingWith(string,Func(string, bool),Order) (Inherited from C1.LiveLinq.Indexing.Index<T>)
	GroupJoin	Overloaded. Correlates the items of this indexed collection with the items of another indexed collection and groups the results by the item of this collection. Implements IIndexScanner(T,TKey).GroupJoin
	Join	Overloaded. Correlates the items of this indexed collection with the items of another sequence and returns the combined items with matching keys. Implements IIndexScanner(T,TKey).Join
	Keys	Gets distinct key values in all items of the indexed collection.

Top

See Also

Reference

Index<T,TKey> Class
C1.LiveLinq.Indexing Namespace

Methods

For a list of all members of this type, see Index<T,TKey> members.

Public Methods

	Name	Description
	All	Gets all items in the indexed collection.
	ContainsKey	Overloaded. Returns a value that indicates whether the collection contains an item with the given key value.
	Find	Overloaded. Finds items with the specified key value.
	FindBetween	Overloaded. Finds items with key values in the interval between the specified values.
	FindGreater	Overloaded. Finds items with keys greater than the specified value.
	FindKeys	Overloaded. Finds items containing any of the specified key values.
	FindLess	Overloaded. Finds items with keys less than the specified value.
	FindSingle	Finds the only item with the specified key value and throws an exception if there is not exactly one item with that key value.
	FindStartingWith	Finds items with string key values starting with the specified string. Implements IIndexScanner(T).FindStartingWith(string,Func(string, bool),Order) (Inherited from C1.LiveLinq.Indexing.Index<T>)
	GroupJoin	Overloaded.

		Correlates the items of this indexed collection with the items of another indexed collection and groups the results by the item of this collection. Implements IIndexScanner(T,TKey).GroupJoin
	Join	Overloaded. Correlates the items of this indexed collection with the items of another sequence and returns the combined items with matching keys. Implements IIndexScanner(T,TKey).Join
	Keys	Gets distinct key values in all items of the indexed collection.

Top

See Also

Reference

Index<T,TKey> Class
C1.LiveLinq.Indexing Namespace

All Method
Specifies the order of the key values to sort the result.

Gets all items in the indexed collection.

Syntax

Visual Basic (Declaration)

```
Public MustOverride Function All( _
   ByVal order As Order _
) As IndexQuery(Of T,TKey)
```

C#

```
public abstract IndexQuery<T,TKey> All(
   Order order
)
```

Parameters

order

Specifies the order of the key values to sort the result.

Return Value

An object enumerating all items of the collection in the specified order of their key values.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T,TKey> Class
Index<T,TKey> Members

ContainsKey Method
Returns a value that indicates whether the collection contains an item with the given key value.

Overload List

Overload	Description
ContainsKey(TKey)	Returns a value that indicates whether the collection contains an item with the given key value.
ContainsKey(Object)	Returns a value that indicates whether the indexed collection contains an item with the given key value. Implements IIndexScanner(T).ContainsKey(object) (Inherited from C1.LiveLinq.Indexing.Index<T>)

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T,TKey> Class
Index<T,TKey> Members

ContainsKey(TKey) Method
The key value to search for.

Returns a value that indicates whether the collection contains an item with the given key value.

Syntax

Visual Basic (Declaration)

`Public Overloads MustOverride Function ContainsKey( _`
`  ByVal key As TKey _`
`) As Boolean`

C#

`public abstract bool ContainsKey(`
`  TKey key`
`)`

Parameters

key

The key value to search for.

Return Value

true if the collection contains an element with the specified key value; otherwise, **false**.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T,TKey> Class
Index<T,TKey> Members
Overload List

Find Method
Finds items with the specified key value.

Overload List

Overload	Description
Find(TKey)	Finds items with the specified key value.
Find(Object)	Finds items with the specified key value. Implements IIndexScanner(T).Find(object) (Inherited from C1.LiveLinq.Indexing.Index<T>)

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T,TKey> Class
Index<T,TKey> Members

Find(TKey) Method
The key value to search for.

Finds items with the specified key value.

Syntax

Visual Basic (Declaration)

`Public Overloads MustOverride Function Find( _`
`  ByVal key As TKey _`
`) As IndexQuery(Of T,TKey)`

C#

`public abstract IndexQuery<T,TKey> Find(`
`  TKey key`
`)`

Parameters

key

The key value to search for.

Return Value

An object enumerating items having the specified key value.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T,TKey> Class
Index<T,TKey> Members
Overload List

FindBetween Method
Finds items with key values in the interval between the specified values.

Overload List

Overload	Description

FindBetween(TKey,Boolean,TKey,Boolean,Func<TKey,Boolean>,Order)	Finds items with key values in the interval between the specified values.
FindBetween(Object,Boolean,Object,Boolean,Order)	Finds items with key values in the interval between the specified values. Implements IIndexScanner(T).FindBetween(object,bool,object,bool,Order) (Inherited from C1.LiveLinq.Indexing.Index<T>)

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T,TKey> Class
Index<T,TKey> Members

FindBetween(TKey,Boolean,TKey,Boolean,Func<TKey,Boolean>,Order) Method
Minimum key value to search for.

If **true**, the result includes items with the minimum key value.

Maximum key value to search for.

If **true**, the result includes items with the maximum key value.

An optional condition that found items must satisfy.

Optionally specifies the order of the key values to sort the result (C1.LiveLinq.Order.**Unordered** if sorting is not required).

Finds items with key values in the interval between the specified values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Overridable Function FindBetween( _
   ByVal min As TKey, _
   ByVal minInclusive As Boolean, _
   ByVal max As TKey, _
   ByVal maxInclusive As Boolean, _
   ByVal keyPredicate As Func(Of TKey,Boolean), _
   ByVal order As Order _
) As IndexQuery(Of T,TKey)
```

C#

```
public virtual IndexQuery<T,TKey> FindBetween(
   TKey min,
   bool minInclusive,
   TKey max,
   bool maxInclusive,
   Func<TKey,bool> keyPredicate,
   Order order
)
```

Parameters

min

Minimum key value to search for.

minInclusive

If **true**, the result includes items with the minimum key value.

max

Maximum key value to search for.

maxInclusive

If **true**, the result includes items with the maximum key value.

keyPredicate

An optional condition that found items must satisfy.

order

Optionally specifies the order of the key values to sort the result (C1.LiveLinq.Order.**Unordered** if sorting is not required).

Return Value

An object enumerating all items with key values within the specified limits.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T,TKey> Class
Index<T,TKey> Members
Overload List

FindGreater Method
Finds items with keys greater than the specified value.

Overload List

Overload	Description
FindGreater(TKey,Boolean,Func<TKey,Boolean>,Order)	Finds items with keys greater than the specified value.
FindGreater(Object,Boolean,Order)	Finds items with keys greater than the specified value. Implements IIndexScanner(T).FindGreater(object,bool,Order) (Inherited from

	C1.LiveLinq.Indexing.Index<T>)

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T,TKey> Class
Index<T,TKey> Members

FindGreater(TKey,Boolean,Func<TKey,Boolean>,Order) Method
Minimum key value to search for.

If **true**, the result includes items with the specified key value. Otherwise, the result only includes those with keys strictly greater than the specified value.

An optional condition that found items must satisfy.

Optionally specifies the order of the key values to sort the result (C1.LiveLinq.Order.**Unordered** if sorting is not required).

Finds items with keys greater than the specified value.

Syntax

Visual Basic (Declaration)

```
Public Overloads Overridable Function FindGreater( _
   ByVal key As TKey, _
   ByVal inclusive As Boolean, _
   ByVal keyPredicate As Func(Of TKey,Boolean), _
   ByVal order As Order _
) As IndexQuery(Of T,TKey)
```

C#

```
public virtual IndexQuery<T,TKey> FindGreater(
   TKey key,
```

```
    bool inclusive,
    Func<TKey,bool> keyPredicate,
    Order order
)
```

Parameters

key

Minimum key value to search for.

inclusive

If **true**, the result includes items with the specified key value. Otherwise, the result only includes those with keys strictly greater than the specified value.

keyPredicate

An optional condition that found items must satisfy.

order

Optionally specifies the order of the key values to sort the result (C1.LiveLinq.Order.**Unordered** if sorting is not required).

Return Value

An object enumerating all items whose key values are greater than the specified value.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T,TKey> Class
Index<T,TKey> Members
Overload List

FindKeys Method
Finds items containing any of the specified key values.

Overload List

Overload	Description
FindKeys(IEnumerable<TKey>,Order)	Finds items containing any of the specified key values.
FindKeys(IEnumerable,Order)	Finds items containing any of the specified key values. Implements FindKeys(IEnumerable,Order) (Inherited from C1.LiveLinq.Indexing.Index<T>)

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T,TKey> Class
Index<T,TKey> Members

FindKeys(IEnumerable<TKey>,Order) Method
The key values to search for.

Optionally specifies the order of the key values to sort the result (C1.LiveLinq.Order.**Unordered** if sorting is not required).

Finds items containing any of the specified key values.

Syntax

Visual Basic (Declaration)

```
Public Overloads MustOverride Function FindKeys( _
   ByVal keys As IEnumerable(Of TKey), _
   ByVal order As Order _
) As IndexQuery(Of T,TKey)
```

C#

```
public abstract IndexQuery<T,TKey> FindKeys(
   IEnumerable<TKey> keys,
   Order order
)
```

Parameters

keys

The key values to search for.

order

Optionally specifies the order of the key values to sort the result (C1.LiveLinq.Order.**Unordered** if sorting is not required).

Return Value

An object enumerating all items whose key values belong to the specified key value collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T,TKey> Class
Index<T,TKey> Members
Overload List

FindLess Method
Finds items with keys less than the specified value.

Overload List

Overload	Description

FindLess(TKey,Boolean,Func<TKey,Boolean>,Order)	Finds items with keys less than the specified value.
FindLess(Object,Boolean,Order)	Finds items with keys less than the specified value. Implements IIndexScanner(T).FindLess(object,bool,Order) (Inherited from C1.LiveLinq.Indexing.Index<T>)

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T,TKey> Class
Index<T,TKey> Members

FindLess(TKey,Boolean,Func<TKey,Boolean>,Order) Method
Maximum key value to search for.

If **true**, the result includes items with the specified key value. Otherwise, the result only includes those with keys strictly less than the specified value.

An optional condition that found items must satisfy.

Optionally specifies the order of the key values to sort the result (C1.LiveLinq.Order.**Unordered** if sorting is not required).

Finds items with keys less than the specified value.

Syntax

Visual Basic (Declaration)	
`Public Overloads Overridable Function FindLess( _`	

```
   ByVal key As TKey, _
   ByVal inclusive As Boolean, _
   ByVal keyPredicate As Func(Of TKey,Boolean), _
   ByVal order As Order _
) As IndexQuery(Of T,TKey)
```

C#

```
public virtual IndexQuery<T,TKey> FindLess(
   TKey key,
   bool inclusive,
   Func<TKey,bool> keyPredicate,
   Order order
)
```

Parameters

key

Maximum key value to search for.

inclusive

If **true**, the result includes items with the specified key value. Otherwise, the result only includes those with keys strictly less than the specified value.

keyPredicate

An optional condition that found items must satisfy.

order

Optionally specifies the order of the key values to sort the result (C1.LiveLinq.Order.**Unordered** if sorting is not required).

Return Value

An object enumerating all items whose key values are less than the specified value.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T,TKey> Class
Index<T,TKey> Members
Overload List

FindSingle Method
The key value to search for.

Finds the only item with the specified key value and throws an exception if there is not exactly one item with that key value.

Syntax

Visual Basic (Declaration)

`Public Overridable Function FindSingle( _`
`  ByVal key As TKey _`
`) As T`

C#

`public virtual T FindSingle(`
`  TKey key`
`)`

Parameters

key

The key value to search for.

Return Value

The item of the collection that contains the specified key value, if found.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T,TKey> Class
Index<T,TKey> Members

GroupJoin Method
Correlates the items of this indexed collection with the items of another indexed collection and groups the results by the item of this collection.

Implements IIndexScanner(T,TKey).GroupJoin

Overload List

Overload	Description
GroupJoin<T2,TResult>(IIndexScanner<T2,TKey>,Func<T,IEnumerable<T2>,TResult>)	Correlates the items of this indexed collection with the items of another indexed collection and groups the results by the item of this collection. Implements IIndexScanner(T,TKey).GroupJoin
GroupJoin<T2,TResult>(IEnumerable<T2>,Func<T2,TKey>,Func<IEnumerable<T>,T2,TResult>)	Correlates the items of this indexed collection with the items of another sequence and groups the results by the item of the second sequence. Implements IIndexScanner(T,TKey).Gr

	oupJoin
GroupJoin<T2,TResult>(IEnumerable<T2>,Func<T2,TKey>,Func<T,IEnumerable<T2>,TResult>)	Correlates the items of this indexed collection with the items of another sequence and groups the results by the item of this collection. Implements IIndexScanner(T,TKey).GroupJoin
GroupJoin<T2,TResult>(IIndexScanner<T2>,Func<T,IEnumerable<T2>,TResult>)	Correlates the items of this indexed collection with the items of another indexed collection and groups the results by the item of this collection. Implements IIndexScanner(T).GroupJoin (Inherited from C1.LiveLinq.Indexing.Index<T>)
GroupJoin<T2,TResult>(IEnumerable<T2>,Func<T2,Object>,Func<IEnumerable<T>,T2,TResult>)	Correlates the items of this indexed collection with the items of

	another sequence and groups the results by the item of the second sequence. Implements IIndexScanner(T).GroupJoin (Inherited from C1.LiveLinq.Indexing.Index<T>)
GroupJoin<T2,TResult>(IEnumerable<T2>,Func<T2,Object>,Func<T,IEnumerable<T2>,TResult>)	Correlates the items of this indexed collection with the items of another sequence and groups the results by the item of this collection. Implements IIndexScanner(T).GroupJoin (Inherited from C1.LiveLinq.Indexing.Index<T>)

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T,TKey> Class
Index<T,TKey> Members

GroupJoin<T2,TResult>(IIndexScanner<T2,TKey>,Func<T,IEnumerable<T2>,TResult>) Method
The type of the elements of the second collection.

The type of the result elements.

The second indexed collection to join to this collection.

A function to create a result element from an element from this collection and a collection of matching elements from the second collection.

Correlates the items of this indexed collection with the items of another indexed collection and groups the results by the item of this collection.

Implements IIndexScanner(T,TKey).GroupJoin

Syntax

Visual Basic (Declaration)

```
Public Overloads Function GroupJoin
    (Of T2,TResult)( _
   ByVal source As IIndexScanner(Of T2,TKey), _
   ByVal resultSelector As Func(Of T,IEnumerable(Of T2),TResult) _
) As IEnumerable(Of TResult)
```

C#

```
public IEnumerable<TResult> GroupJoin<T2,TResult>(
   IIndexScanner<T2,TKey> source,
   Func<T,IEnumerable<T2>,TResult> resultSelector
)
```

Parameters

source

The second indexed collection to join to this collection.

resultSelector

A function to create a result element from an element from this collection and a collection of matching elements from the second collection.

Type Parameters

T2

The type of the elements of the second collection.

TResult

The type of the result elements.

Return Value

Enumeration of objects obtained by applying the result selector to group pairs, where each pair consists of an item of this collection and the corresponding enumeration of the items of the second collection joined to it.

Remarks

Matching of two elements is performed by matching their keys.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T,TKey> Class
Index<T,TKey> Members
Overload List

GroupJoin<T2,TResult>(IEnumerable<T2>,Func<T2,TKey>,Func<IEnumerable<T>,T2,TResult>) Method
The type of the elements of the second sequence.

The type of the result elements.

The second sequence to join to this collection.

A function to extract from an item of the second sequence the value to match against this collection's key value.

A function to create a result element from an element of the second sequence and the collection of matching elements from this collection.

Correlates the items of this indexed collection with the items of another sequence and groups the results by the item of the second sequence.

Implements IIndexScanner(T,TKey).GroupJoin

Syntax

Visual Basic (Declaration)

```
Public Overloads Function GroupJoin
    (Of T2,TResult)( _
   ByVal source As IEnumerable(Of T2), _
   ByVal keySelector As Func(Of T2,TKey), _
   ByVal resultSelector As Func(Of IEnumerable(Of T),T2,TResult) _
) As IEnumerable(Of TResult)
```

C#

```
public IEnumerable<TResult> GroupJoin<T2,TResult>(
   IEnumerable<T2> source,
   Func<T2,TKey> keySelector,
   Func<IEnumerable<T>,T2,TResult> resultSelector
)
```

Parameters

source

The second sequence to join to this collection.

keySelector

A function to extract from an item of the second sequence the value to match against this collection's key value.

resultSelector

A function to create a result element from an element of the second sequence and the collection of matching elements from this collection.

Type Parameters

T2

The type of the elements of the second sequence.

TResult

The type of the result elements.

Return Value

Enumeration of objects obtained by applying the result selector to group pairs, where each pair consists of an item of the second collection and the corresponding enumeration of the items of this collection joined to it.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T,TKey> Class
Index<T,TKey> Members
Overload List

GroupJoin<T2,TResult>(IEnumerable<T2>,Func<T2,TKey>,Func<T,IEnumerable<T2>,TResult>) Method
The type of the elements of the second sequence.

The type of the result elements.

The second sequence to join to this collection.

A function to extract from an item of the second sequence the value to match against this collection's key value.

A function to create a result element from an element from this collection and a collection of matching elements from the second sequence.

Correlates the items of this indexed collection with the items of another sequence and groups the results by the item of this collection.

Implements IIndexScanner(T,TKey).GroupJoin

Syntax

Visual Basic (Declaration)

```
Public Overloads Function GroupJoin
    (Of T2,TResult)( _
   ByVal source As IEnumerable(Of T2), _
   ByVal keySelector As Func(Of T2,TKey), _
   ByVal resultSelector As Func(Of T,IEnumerable(Of T2),TResult) _
) As IEnumerable(Of TResult)
```

C#

```
public IEnumerable<TResult> GroupJoin<T2,TResult>(
   IEnumerable<T2> source,
   Func<T2,TKey> keySelector,
   Func<T,IEnumerable<T2>,TResult> resultSelector
)
```

Parameters

source

The second sequence to join to this collection.

keySelector

A function to extract from an item of the second sequence the value to match against this collection's key value.

resultSelector

A function to create a result element from an element from this collection and a collection of matching elements from the second sequence.

Type Parameters

T2

The type of the elements of the second sequence.

TResult

The type of the result elements.

Return Value

Enumeration of objects obtained by applying the result selector to group pairs, where each pair consists of an item of this collection and

the corresponding enumeration of the items of the second sequence joined to it.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T,TKey> Class
Index<T,TKey> Members
Overload List

Join Method
Correlates the items of this indexed collection with the items of another sequence and returns the combined items with matching keys.

Implements IIndexScanner(T,TKey).Join

Overload List

Overload	Description
Join<T2,TResult>(IEnumerable<T2>,Func<T2,TKey>,Func<T,T2,TResult>,JoinOperator)	Correlates the items of this indexed collection with the items of another sequence and returns the combined items with matching keys. Implements IIndexScanner(T,TKey).Join

Join<T2,TResult>(IIndexScanner<T2,TKey>,Func<T,T2,TResult>,JoinOperator)	Correlates the items of this indexed collection with the items of another indexed collection and returns the combined items with matching keys. Implements IIndexScanner(T,TKey).Join
Join<T2,TResult>(IEnumerable<T2>,Func<T2,Object>,Func<T,T2,TResult>,JoinOperator)	Correlates the items of this indexed collection with the items of another sequence and returns the combined items with matching keys. Implements IIndexScanner(T).Join (Inherited from C1.LiveLinq.Indexing.Index<T>)
Join<T2,TResult>(IIndexScanner<T2>,Func<T,T2,TResult>,JoinOperator)	Correlates the items of this indexed collection with the items of another indexed collection and returns the combined items with matching keys. Implements IIndexScanner(T).Join

	(Inherited from C1.LiveLinq.Indexing.Index<T>)

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T,TKey> Class
Index<T,TKey> Members

Join<T2,TResult>(IEnumerable<T2>,Func<T2,TKey>,Func<T,T2,TResult>,JoinOperator) Method
The type of the elements of the second sequence.

The type of the result elements.

The second sequence to join to this collection.

A function to extract from a second sequence's item the value to match against this collection's key value.

A function to create a result element from two matching elements.

A comparison operator to match elements.

Correlates the items of this indexed collection with the items of another sequence and returns the combined items with matching keys.

Implements IIndexScanner(T,TKey).Join

Syntax

Visual Basic (Declaration)	

```
Public Overloads Function Join
    (Of T2,TResult)( _
   ByVal source As IEnumerable(Of T2), _
   ByVal keySelector As Func(Of T2,TKey), _
```

```
   ByVal resultSelector As Func(Of T,T2,TResult), _
   ByVal op As JoinOperator _
) As IEnumerable(Of TResult)
```

C#

```
public IEnumerable<TResult> Join<T2,TResult>(
   IEnumerable<T2> source,
   Func<T2,TKey> keySelector,
   Func<T,T2,TResult> resultSelector,
   JoinOperator op
)
```

Parameters

source

The second sequence to join to this collection.

keySelector

A function to extract from a second sequence's item the value to match against this collection's key value.

resultSelector

A function to create a result element from two matching elements.

op

A comparison operator to match elements.

Type Parameters

T2

The type of the elements of the second sequence.

TResult

The type of the result elements.

Return Value

Enumeration of objects obtained by applying the result selector to pairs of joined elements of the two collections.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T,TKey> Class
Index<T,TKey> Members
Overload List

Join<T2,TResult>(IIndexScanner<T2,TKey>,Func<T,T2,TResult>,JoinOperator) Method

The type of the elements of the second collection.

The type of the result elements.

The second indexed collection to join to this collection.

A function to create a result element from two matching elements.

A comparison operator to match elements.

Correlates the items of this indexed collection with the items of another indexed collection and returns the combined items with matching keys.

Implements IIndexScanner(T,TKey).Join

Syntax

Visual Basic (Declaration)

```
Public Overloads Function Join
    (Of T2,TResult)( _
   ByVal source As IIndexScanner(Of T2,TKey), _
   ByVal resultSelector As Func(Of T,T2,TResult), _
   ByVal op As JoinOperator _
) As IEnumerable(Of TResult)
```

C#

```
public IEnumerable<TResult> Join<T2,TResult>(
   IIndexScanner<T2,TKey> source,
   Func<T,T2,TResult> resultSelector,
   JoinOperator op
)
```

Parameters

source

The second indexed collection to join to this collection.

resultSelector

A function to create a result element from two matching elements.

op

A comparison operator to match elements.

Type Parameters

T2

The type of the elements of the second collection.

TResult

The type of the result elements.

Return Value

Enumeration of objects obtained by applying the result selector to pairs of joined elements of the two collections.

Remarks

Matching of two elements is performed by matching their keys.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T,TKey> Class
Index<T,TKey> Members
Overload List

Keys Method

Specifies the order of the key values to sort the result.

Gets distinct key values in all items of the indexed collection.

Syntax

Visual Basic (Declaration)

```
Public MustOverride Function Keys( _
   ByVal order As Order _
) As IEnumerable(Of TKey)
```

C#

```
public abstract IEnumerable<TKey> Keys(
   Order order
)
```

Parameters

order

Specifies the order of the key values to sort the result.

Return Value

All distinct key values contained in the items of the collection in the specified order.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T,TKey> Class
Index<T,TKey> Members

Properties
For a list of all members of this type, see Index<T,TKey> members.

Public Properties

	Name	Description
	Algorithm	Gets the indexing algorithm used by the index. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	ItemCount	Gets the number of elements in the indexed collection. (Inherited from C1.LiveLinq.Indexing.Index<T>)
	KeyCount	Gets the number of distinct key values in all items of this collection. (Inherited from C1.LiveLinq.Indexing.Index<T>)
	KeyIsUnique	Gets a value that indicates whether the key used in this index is a unique key for the collection. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	KeySelector	Gets the expression used to obtain key value from an element of the indexed collection.
	KeyType	Gets the type of the index key. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	Locale	Gets the locale information used to compare strings in the index. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	Root	Gets the root index in an index/subindex hierarchy. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	Subindexes	Gets the collection of subindexes added to this index. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)

Top

See Also

Reference

Index<T,TKey> Class
C1.LiveLinq.Indexing Namespace

KeySelector Property
Gets the expression used to obtain key value from an element of the indexed collection.

Syntax

Visual Basic (Declaration)
`Public Shadows ReadOnly Property KeySelector As Expression(Of Func(Of T,TKey))`
C#
`public new Expression<Func<T,TKey>> KeySelector {get;}`

Property Value

An expression calculating the key value from an item (element of the collection). Typically, this is a field or a property in the item class, although more complex expressions can also be used.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Index<T,TKey> Class
Index<T,TKey> Members

IndexCollection<T>
The type of the elements of the indexed collection.

Represents a collection of indexes attached to an indexed collection.

Object Model

IndexCollection<T>

Syntax

Visual Basic (Declaration)
`Public Class IndexCollection(Of T)` `  Inherits ScannerCollection(Of T)`
C#
`public class IndexCollection<T> : ScannerCollection<T>`

Type Parameters

T

The type of the elements of the indexed collection.

Remarks

Any indexed collection (implementing the IIndexedSource<T> interface) has a collection of indexes attached to it.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.Indexing.ScannerCollection<T>
    C1.LiveLinq.Indexing.IndexCollection<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexCollection<T> Members
C1.LiveLinq.Indexing Namespace
Indexes Property

Indexes Property
Indexes Property
Indexes Property
Indexes Property

Overview
The type of the elements of the indexed collection.

Represents a collection of indexes attached to an indexed collection.

Object Model

Syntax

Visual Basic (Declaration)
`Public Class IndexCollection(Of T)` `  Inherits ScannerCollection(Of T)`
C#
`public class IndexCollection<T> : ScannerCollection<T>`

Type Parameters

T

The type of the elements of the indexed collection.

Remarks

Any indexed collection (implementing the IIndexedSource<T> interface) has a collection of indexes attached to it.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.Indexing.ScannerCollection<T>
    C1.LiveLinq.Indexing.IndexCollection<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexCollection<T> Members
C1.LiveLinq.Indexing Namespace
Indexes Property
Indexes Property
Indexes Property
Indexes Property
Indexes Property

Members

Properties Methods

The following tables list the members exposed by IndexCollection<T>.

Public Constructors

	Name	Description
	IndexCollection<T> Constructor	Initializes a new instance of the IndexCollection<T> class.

Top

Public Properties

	Name	Description
	Count	Overridden. Gets the number of indexes in the collection.
	Item	Gets the index with the specified key selector.

Top

Public Methods

	Name	Description
	Add	Overloaded. Creates a new index and adds it to the collection of indexes.
	Clear	Overridden. Clears the collection of all indexes. All indexes are detached from the indexed collection and destroyed.
	Contains	Overloaded. Determines whether an index with the specified key selector exists in the collection. (Inherited from C1.LiveLinq.Indexing.ScannerCollection<T>)
	Find	Overloaded. Finds an index in the collection by its key selector.
	GetEnumerator	Overridden. Returns an enumerator that iterates through the IndexCollection<T>.
	Remove	Overloaded. Overridden. Removes an index from the collection.

Top

See Also

Reference

IndexCollection<T> Class
C1.LiveLinq.Indexing Namespace
Indexes Property
Indexes Property
Indexes Property
Indexes Property
Indexes Property

IndexCollection<T> Constructor
The collection to be indexed.

Initializes a new instance of the IndexCollection<T> class.

Syntax

Visual Basic (Declaration)

```
Public Function New( _
   ByVal source As IObservableSource(Of T) _
)
```

C#

```
public IndexCollection<T>(
   IObservableSource<T> source
)
```

Parameters

source

The collection to be indexed.

Remarks

Normally, you don't need to create instances of the IndexCollection<T> class in your code, they already exist in the instances of LiveLinq indexed collection classes. You need to create them explicitly in code only if you define your own indexable collection class and then only if it does not inherit from C1.LiveLinq.Collections.IndexedCollection<T>.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexCollection<T> Class
IndexCollection<T> Members

Methods
For a list of all members of this type, see IndexCollection<T> members.

Public Methods

	Name	Description
	Add	Overloaded. Creates a new index and adds it to the collection of indexes.
	Clear	Overridden. Clears the collection of all indexes. All indexes are detached from the indexed collection and destroyed.
	Contains	Overloaded. Determines whether an index with the specified key selector exists in the collection. (Inherited from C1.LiveLinq.Indexing.ScannerCollection<T>)
	Find	Overloaded. Finds an index in the collection by its key selector.
	GetEnumerator	Overridden. Returns an enumerator that iterates through the IndexCollection<T>.
	Remove	Overloaded. Overridden. Removes an index from the collection.

Top

See Also

Reference

IndexCollection<T> Class
C1.LiveLinq.Indexing Namespace
Indexes Property
Indexes Property
Indexes Property
Indexes Property
Indexes Property

Add Method
Creates a new index and adds it to the collection of indexes.

Overload List

Overload	Description

Add<TKey>(Expression<Func<T,TKey>>)	Creates a new index and adds it to the collection of indexes.
Add<TKey>(Expression<Func<T,TKey>>,Boolean)	Creates a new index and adds it to the collection of indexes.
Add<TKey>(Expression<Func<T,TKey>>,Boolean,Boolean,IndexingAlgorithm,CultureInfo)	Creates a new index and adds it to the collection of indexes. (Inherited from C1.LiveLinq.Indexing.ScannerCollection<T>)
Add<TKey>(Expression<Func<T,TKey>>,Boolean,Boolean,CultureInfo)	Creates a new index and adds it to the collection of indexes. (Inherited from C1.LiveLinq.Indexing.ScannerCollection<T>)
Add<TKey>(Expression<Func<T,TKey>>,Boolean,Boolean)	Creates a new index and adds it to the collection of indexes. (Inherited from C1.LiveLinq.Indexing.ScannerCollection<T>)
Add(LambdaExpression,Boolean,Boolean,IndexingAlgorithm,CultureInfo)	Creates a new index and adds it to the collection of indexes. (Inherited from C1.LiveLinq.Indexing.ScannerCollection<T>)

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexCollection<T> Class
IndexCollection<T> Members

Add<TKey>(Expression<Func<T,TKey>>) Method
The type of the index key.

Key selector expression of the index, see IndexDefinition<T>.KeySelector.

Creates a new index and adds it to the collection of indexes.

Syntax

Visual Basic (Declaration)

`Public Overloads Function Add(Of TKey)( _`
`  ByVal keySelector As Expression(Of Func(Of T,TKey)) _`
`) As Index(Of T,TKey)`

C#

`public Index<T,TKey> Add<TKey>(`
`  Expression<Func<T,TKey>> keySelector`
`)`

Parameters

keySelector

Key selector expression of the index, see IndexDefinition<T>.KeySelector.

Type Parameters

TKey

The type of the index key.

Return Value

The new index added to the collection of indexes.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexCollection<T> Class
IndexCollection<T> Members
Overload List

Add<TKey>(Expression<Func<T,TKey>>,Boolean) Method
The type of the index key.

Key selector expression of the index, see IndexDefinition<T>.KeySelector.

true if a unique index must be created.

Creates a new index and adds it to the collection of indexes.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function Add(Of TKey)( _
   ByVal keySelector As Expression(Of Func(Of T,TKey)), _
   ByVal keyIsUnque As Boolean _
) As Index(Of T,TKey)
```

C#

```
public Index<T,TKey> Add<TKey>(
   Expression<Func<T,TKey>> keySelector,
   bool keyIsUnque
)
```

Parameters

keySelector

Key selector expression of the index, see IndexDefinition<T>.KeySelector.

keyIsUnque

true if a unique index must be created.

Type Parameters

TKey

The type of the index key.

Return Value

The new index added to the collection of indexes.

Remarks

A unique index occupies less memory and performs better than a non-unique index (although the difference isn't dramatic). Therefore, for unique keys, it's recommended to specify the corresponding index as unique.

But do that only if you are sure that the key is indeed unique, as it imposes a uniqueness constraint on the indexed collection. An attempt to modify the indexed collection violating the uniqueness throws an System.InvalidOperationException.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexCollection<T> Class
IndexCollection<T> Members
Overload List

Clear Method

Clears the collection of all indexes. All indexes are detached from the indexed collection and destroyed.

Syntax

Visual Basic (Declaration)	
`Public Overrides Sub Clear()`	

C#
`public override void Clear()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexCollection<T> Class
IndexCollection<T> Members

Find Method
Finds an index in the collection by its key selector.

Overload List

Overload	Description
Find(LambdaExpression)	Finds an index in the collection by its key selector.
Find<TKey>(Expression<Func<T,TKey>>)	Finds an index in the collection by its key selector.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexCollection<T> Class
IndexCollection<T> Members

Find(LambdaExpression) Method
Key selector expression of an index, see IndexDefinition<T>.KeySelector.

Finds an index in the collection by its key selector.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shadows Function Find( _
   ByVal keySelector As LambdaExpression _
) As Index(Of T)
```

C#

```
public new Index<T> Find(
   LambdaExpression keySelector
)
```

Parameters

keySelector

Key selector expression of an index, see IndexDefinition<T>.KeySelector.

Return Value

An index with the given key selector, if it is found; otherwise, **null**.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexCollection<T> Class
IndexCollection<T> Members
Overload List

Find<TKey>(Expression<Func<T,TKey>>) Method
The type of the index key.

Key selector expression of an index, see IndexDefinition<T>.KeySelector.

Finds an index in the collection by its key selector.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function Find(Of TKey)( _
   ByVal keySelector As Expression(Of Func(Of T,TKey)) _
) As Index(Of T,TKey)
```

C#

```
public Index<T,TKey> Find<TKey>(
   Expression<Func<T,TKey>> keySelector
)
```

Parameters

keySelector

Key selector expression of an index, see IndexDefinition<T>.KeySelector.

Type Parameters

TKey

The type of the index key.

Return Value

An index with the given key selector, if it is found; otherwise, **null**.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexCollection<T> Class
IndexCollection<T> Members
Overload List

GetEnumerator Method
Returns an enumerator that iterates through the IndexCollection<T>.

Syntax

Visual Basic (Declaration)
`Public Overrides Function GetEnumerator() As IEnumerator(Of IIndexScanner(Of T))`
C#
`public override IEnumerator<IIndexScanner<T>> GetEnumerator()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexCollection<T> Class
IndexCollection<T> Members

Remove Method
Removes an index from the collection.

Overload List

Overload	Description
Remove(LambdaExpression)	Removes an index from the collection.
Remove(Index<T>)	Removes an index from the collection.
Remove<TKey>(Expression<Func<T,TKey>>)	Removes an index from the collection. (Inherited from C1.LiveLinq.Indexing.ScannerCollection<T>)

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexCollection<T> Class
IndexCollection<T> Members

Remove(LambdaExpression) Method
Key selector expression of an index, see IndexDefinition<T>.KeySelector.

Removes an index from the collection.

Syntax

Visual Basic (Declaration)

```
Public Overloads Overrides Function Remove( _
   ByVal keySelector As LambdaExpression _
) As Boolean
```

C#

```
public override bool Remove(
   LambdaExpression keySelector
)
```

Parameters

keySelector

Key selector expression of an index, see IndexDefinition<T>.KeySelector.

Return Value

true if an index has been removed; **false** if there is no index with the given key selector in the collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexCollection<T> Class
IndexCollection<T> Members
Overload List

Remove(Index<T>) Method
The index to remove.

Removes an index from the collection.

Syntax

Visual Basic (Declaration)

`Public Overloads Function Remove( _`
`  ByVal index As Index(Of T) _`
`) As Boolean`

C#

`public bool Remove(`
`  Index<T> index`
`)`

Parameters

index

The index to remove.

Return Value

true if an index has been removed; **false** if the index does not belong to this collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexCollection<T> Class
IndexCollection<T> Members
Overload List

Properties
For a list of all members of this type, see IndexCollection<T> members.

Public Properties

	Name	Description
	Count	Overridden. Gets the number of indexes in the collection.
	Item	Gets the index with the specified key selector.

Top

See Also

Reference

IndexCollection<T> Class
C1.LiveLinq.Indexing Namespace
Indexes Property
Indexes Property
Indexes Property
Indexes Property
Indexes Property

Count Property
Gets the number of indexes in the collection.

Syntax

Visual Basic (Declaration)

```
Public Overrides ReadOnly Property Count As Integer
```

C#

```
public override int Count {get;}
```

Property Value

The number of indexes in the collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexCollection<T> Class
IndexCollection<T> Members

Item Property
Key selector expression of an index, see IndexDefinition<T>.KeySelector.

Gets the index with the specified key selector.

Syntax

Visual Basic (Declaration)

`Public ReadOnly Default Property Item( _`
`  ByVal key As LambdaExpression _`
`) As Index(Of T)`

C#

`public Index<T> this[`
`  LambdaExpression key`
`]; {get;}`

Parameters

key

Key selector expression of an index, see IndexDefinition<T>.KeySelector.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

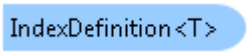
Reference

IndexCollection<T> Class
IndexCollection<T> Members

IndexDefinition<T>
The type of the elements of indexed collection.

Contains common part of the Index and Subindex classes.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit Class IndexDefinition(Of T)`
C#
`public abstract class IndexDefinition<T>`

Type Parameters

T

The type of the elements of indexed collection.

Remarks

This class serves as the base class of two classes: Index and Subindex. It contains properties common to these two classes.

Inheritance Hierarchy

System.Object
C1.LiveLinq.Indexing.IndexDefinition<T>
C1.LiveLinq.Indexing.Index<T>
C1.LiveLinq.Indexing.Subindex<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

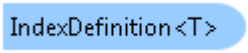
Reference

IndexDefinition<T> Members
C1.LiveLinq.Indexing Namespace
Index<T,TKey> Class

Overview
The type of the elements of indexed collection.

Contains common part of the Index and Subindex classes.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit Class IndexDefinition(Of T)`
C#
`public abstract class IndexDefinition<T>`

Type Parameters

T

The type of the elements of indexed collection.

Remarks

This class serves as the base class of two classes: Index and Subindex. It contains properties common to these two classes.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.Indexing.IndexDefinition<T>
    C1.LiveLinq.Indexing.Index<T>
    C1.LiveLinq.Indexing.Subindex<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexDefinition<T> Members
C1.LiveLinq.Indexing Namespace
Index<T,TKey> Class

Members
Properties

The following tables list the members exposed by IndexDefinition<T>.

Public Properties

	Name	Description
	Algorithm	Gets the indexing algorithm used by the index.
	KeyIsUnique	Gets a value that indicates whether the key used in this index is a unique key for the collection.
	KeySelector	Gets the expression used to obtain key value from an element of the indexed collection.
	KeyType	Gets the type of the index key.

	Locale	Gets the locale information used to compare strings in the index.
	Root	Gets the root index in an index/subindex hierarchy.
	Subindexes	Gets the collection of subindexes added to this index.

Top

See Also

Reference

IndexDefinition<T> Class
C1.LiveLinq.Indexing Namespace
Index<T,TKey> Class

Properties
For a list of all members of this type, see IndexDefinition<T> members.

Public Properties

	Name	Description
	Algorithm	Gets the indexing algorithm used by the index.
	KeyIsUnique	Gets a value that indicates whether the key used in this index is a unique key for the collection.
	KeySelector	Gets the expression used to obtain key value from an element of the indexed collection.
	KeyType	Gets the type of the index key.
	Locale	Gets the locale information used to compare strings in the index.
	Root	Gets the root index in an index/subindex hierarchy.
	Subindexes	Gets the collection of subindexes added to this index.

Top

See Also

Reference

IndexDefinition<T> Class
C1.LiveLinq.Indexing Namespace
Index<T,TKey> Class

Algorithm Property
Gets the indexing algorithm used by the index.

Syntax

Visual Basic (Declaration)
`Public MustOverride ReadOnly Property Algorithm As IndexingAlgorithm`
C#
`public abstract IndexingAlgorithm Algorithm {get;}`

Property Value

An IndexingAlgorithm used by the index. In the current version, only one algorithm is supported, RedBlackTree. Later versions may support other algorithms, such as bitmap or hash indexes.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexDefinition<T> Class
IndexDefinition<T> Members

KeyIsUnique Property
Gets a value that indicates whether the key used in this index is a unique key for the collection.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property KeyIsUnique As Boolean`
C#
`public bool KeyIsUnique {get;}`

Property Value

true if the key is unique; otherwise, **false**

Remarks

A unique index occupies less memory and performs better than a non-unique index (although the difference isn't dramatic). Therefore, for unique keys, it's recommended to specify the corresponding index as unique in the IndexCollection.Add method.

But do that only if you are sure that the key is indeed unique, as it imposes a uniqueness constraint on the collection. An attempt to modify the collection violating the uniqueness throws an System.InvalidOperationException.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexDefinition<T> Class
IndexDefinition<T> Members

KeySelector Property
Gets the expression used to obtain key value from an element of the indexed collection.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property KeySelector As LambdaExpression`

C#
`public LambdaExpression KeySelector {get;}`

Property Value

An expression calculating the key value from an item (element of the collection). Typically, this is a field or a property in the item class, although more complex expressions can also be used.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexDefinition<T> Class
IndexDefinition<T> Members

KeyType Property
Gets the type of the index key.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property KeyType As Type`
C#
`public Type KeyType {get;}`

Property Value

The type of the result of the KeySelector expression.

Remarks

This type is the same as the TKey type parameter for Index<T,TKey> and Subindex<T,TKey> objects.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexDefinition<T> Class
IndexDefinition<T> Members

Locale Property
Gets the locale information used to compare strings in the index.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Locale As CultureInfo`
C#
`public CultureInfo Locale {get;}`

Property Value

A System.Globalization.CultureInfo that contains data about the user's locale. The default is System.Globalization.CultureInfo.CurrentCulture.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexDefinition<T> Class
IndexDefinition<T> Members

Root Property
Gets the root index in an index/subindex hierarchy.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Property Root As IndexDefinition(Of T)
```

C#

```
public IndexDefinition<T> Root {get;}
```

Property Value

For a subindex, this is an Index<T> to which the subindex belongs (maybe indirectly, through several levels of parent subindexes). For an index, this is always the same object as the index itself.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexDefinition<T> Class
IndexDefinition<T> Members

Subindexes Property
Gets the collection of subindexes added to this index.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Property Subindexes As SubindexCollection(Of T)
```

C#

```
public SubindexCollection<T> Subindexes {get;}
```

Property Value

The SubindexCollection<T> that contains the subindexes of this index.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexDefinition<T> Class
IndexDefinition<T> Members

IndexingAlgorithm
Defines the kind of an index, the algorithm used by that index. Currently, the RedBlackTree algorithm is always used.

Object Model

Syntax

Visual Basic (Declaration)
`Public MustInherit Class IndexingAlgorithm`
C#
`public abstract class IndexingAlgorithm`

Remarks

In the current version, only one algorithm is supported, RedBlackTree. Later versions may support other algorithms, such as bitmap or hash indexing.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.Indexing.IndexingAlgorithm

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

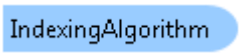
See Also

Reference

IndexingAlgorithm Members
C1.LiveLinq.Indexing Namespace

Overview
Defines the kind of an index, the algorithm used by that index. Currently, the RedBlackTree algorithm is always used.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit Class IndexingAlgorithm`
C#
`public abstract class IndexingAlgorithm`

Remarks

In the current version, only one algorithm is supported, RedBlackTree. Later versions may support other algorithms, such as bitmap or hash indexing.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.Indexing.IndexingAlgorithm

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexingAlgorithm Members
C1.LiveLinq.Indexing Namespace

Members
Fields Methods

The following tables list the members exposed by IndexingAlgorithm.

Public Fields

	Name	Description
S	RedBlackTree	The red-black tree algorithm, a type of self-balancing binary search tree widely used in computer science because it has good performance that does not significantly degrade even in worst cases.

Top

Public Methods

	Name	Description
	CreateIndex<T,TKey>	Creates a new index.

Top

See Also

Reference

IndexingAlgorithm Class
C1.LiveLinq.Indexing Namespace

Methods
For a list of all members of this type, see IndexingAlgorithm members.

Public Methods

	Name	Description
	CreateIndex<T,TKey>	Creates a new index.

Top

See Also

Reference

IndexingAlgorithm Class
C1.LiveLinq.Indexing Namespace

CreateIndex<T,TKey> Method
The type of the elements of the collection to index.

The type of the index key.

The collection to be indexed.

Key selector expression of the index, see KeySelector.

Specifies whether the key used in this index is a unique key for the indexed collection.

Locale information used to compare strings in the index (default: CultureInfo.CurrentCulture).

Creates a new index.

Syntax

Visual Basic (Declaration)

```
Public Function CreateIndex
    (Of T,TKey)( _
   ByVal source As IObservableSource(Of T), _
   ByVal keySelector As Expression(Of Func(Of T,TKey)), _
   ByVal keyIsUnique As Boolean, _
   ByVal locale As CultureInfo _
) As Index(Of T,TKey)
```

C#

```
public Index<T,TKey> CreateIndex<T,TKey>(
   IObservableSource<T> source,
```

```
    Expression<Func<T,TKey>> keySelector,
    bool keyIsUnique,
    CultureInfo locale
)
```

Parameters

source

The collection to be indexed.

keySelector

Key selector expression of the index, see KeySelector.

keyIsUnique

Specifies whether the key used in this index is a unique key for the indexed collection.

locale

Locale information used to compare strings in the index (default: CultureInfo.CurrentCulture).

Type Parameters

T

The type of the elements of the collection to index.

TKey

The type of the index key.

Return Value

The new index.

Remarks

Normally, you don't need to use this method. Indexes are usually created in code by calling IndexCollection.Add, or their creation can be enforced in LINQ queries by using the Indexed hint. This method should be used only in special situations where all you want is to index a collection without an overhead of maintaining a collection of indexes, or you need an index that exists separately from the collection of indexes maintained by the source. Unlike the standard ways of creating indexes, this method only creates an

index object and attaches it to the source (so it will be automatically synchronized with the source when the source is modified), but the created index is not added to IndexCollection<T>.

A unique index occupies less memory and performs better than a non-unique index (although the difference isn't dramatic). Therefore, for unique keys, it's recommended to specify the corresponding index as unique. But do that only if you are sure that the key is indeed unique, as it imposes a uniqueness constraint on the indexed collection. An attempt to modify the indexed collection violating the uniqueness throws an System.InvalidOperationException.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexingAlgorithm Class
IndexingAlgorithm Members

Fields
For a list of all members of this type, see IndexingAlgorithm members.

Public Fields

	Name	Description
S	RedBlackTree	The red-black tree algorithm, a type of self-balancing binary search tree widely used in computer science because it has good performance that does not significantly degrade even in worst cases.

Top

See Also

Reference

IndexingAlgorithm Class
C1.LiveLinq.Indexing Namespace

RedBlackTree Field
The red-black tree algorithm, a type of self-balancing binary search tree widely used in computer science because it has good performance that does not significantly degrade even in worst cases.

Syntax

Visual Basic (Declaration)	
`Public Shared ReadOnly RedBlackTree As IndexingAlgorithm`	
C#	
`public static readonly IndexingAlgorithm RedBlackTree`	

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

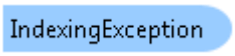
See Also

Reference

IndexingAlgorithm Class
IndexingAlgorithm Members

IndexingException
Represents an exception that is thrown when errors are generated using LiveLinq components.

Object Model



Syntax

Visual Basic (Declaration)	

```
Public Class IndexingException
   Inherits System.Exception
```

C#

```
public class IndexingException : System.Exception
```

Inheritance Hierarchy

System.Object
  System.Exception
    C1.LiveLinq.Indexing.IndexingException

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

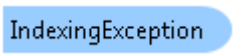
See Also

Reference

IndexingException Members
C1.LiveLinq.Indexing Namespace

Overview
Represents an exception that is thrown when errors are generated using LiveLinq components.

Object Model



Syntax

Visual Basic (Declaration)

```
Public Class IndexingException
   Inherits System.Exception
```

C#

```
public class IndexingException : System.Exception
```

Inheritance Hierarchy

System.Object
  System.Exception
    C1.LiveLinq.Indexing.IndexingException

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexingException Members
C1.LiveLinq.Indexing Namespace

Members
Properties Methods

The following tables list the members exposed by IndexingException.

Public Constructors

	Name	Description
	IndexingException Constructor	Overloaded.

Top

Public Properties

	Name	Description
	Data	(Inherited from System.Exception)
	HelpLink	(Inherited from System.Exception)
	InnerException	(Inherited from System.Exception)

	Message	(Inherited from System.Exception)
	Source	(Inherited from System.Exception)
	StackTrace	(Inherited from System.Exception)
	TargetSite	(Inherited from System.Exception)

Top

Public Methods

	Name	Description
	GetBaseException	(Inherited from System.Exception)
	GetObjectData	(Inherited from System.Exception)
	GetType	(Inherited from System.Exception)
	ToString	(Inherited from System.Exception)

Top

See Also

Reference

IndexingException Class
C1.LiveLinq.Indexing Namespace

IndexingException Constructor

Overload List

Overload	Description
IndexingException Constructor()	Initializes a new instance of the IndexingException class. This is the default constructor.

IndexingException Constructor(String)	Initializes a new instance of the IndexingException class with the specified string.
IndexingException Constructor(String,Exception)	Initializes a new instance of the IndexingException class with the specified string and inner exception.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexingException Class
IndexingException Members

IndexingException Constructor()
Initializes a new instance of the IndexingException class. This is the default constructor.

Syntax

Visual Basic (Declaration)

```
Public Function New()
```

C#

```
public IndexingException()
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexingException Class
IndexingException Members
Overload List

IndexingException Constructor(String)
The string to display when the exception is thrown.

Initializes a new instance of the IndexingException class with the specified string.

Syntax

Visual Basic (Declaration)

`Public Function New( _`
`  ByVal message As String _`
`)`

C#

`public IndexingException(`
`  string message`
`)`

Parameters

message

The string to display when the exception is thrown.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexingException Class
IndexingException Members
Overload List

IndexingException Constructor(String,Exception)
The string to display when the exception is thrown.

A reference to an inner exception.

Initializes a new instance of the IndexingException class with the specified string and inner exception.

Syntax

Visual Basic (Declaration)

```
Public Function New( _
   ByVal message As String, _
   ByVal inner As Exception _
)
```

C#

```
public IndexingException(
   string message,
   Exception inner
)
```

Parameters

message

The string to display when the exception is thrown.

inner

A reference to an inner exception.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

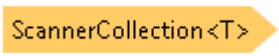
Reference

IndexingException Class
IndexingException Members
Overload List

ScannerCollection<T>
The type of the elements of the indexed collection.

Represents a collection of indexes or subindexes.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit Class ScannerCollection(Of T)`
C#
`public abstract class ScannerCollection<T>`

Type Parameters

T

The type of the elements of the indexed collection.

Remarks

Any indexed collection (implementing the IIndexedSource<T> interface) has a collection of indexes attached to it. **ScannerCollection<T>** is the base class for the collection of indexes, IndexCollection<T>

The ScannerCollection<T> class is also used in the Indexes property of the C1.LiveLinq.Indexing.Search.IndexQuery<T> class, the result of an indexing search operation such as Index.Find and others, where it contains subindexes.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.Indexing.ScannerCollection<T>
    C1.LiveLinq.Indexing.IndexCollection<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ScannerCollection<T> Members
C1.LiveLinq.Indexing Namespace

Overview
The type of the elements of the indexed collection.

Represents a collection of indexes or subindexes.

Object Model

Syntax

Visual Basic (Declaration)
`Public MustInherit Class ScannerCollection(Of T)`
C#
`public abstract class ScannerCollection<T>`

Type Parameters

T

The type of the elements of the indexed collection.

Remarks

Any indexed collection (implementing the IIndexedSource<T> interface) has a collection of indexes attached to it. **ScannerCollection<T>** is the base class for the collection of indexes, IndexCollection<T>

The ScannerCollection<T> class is also used in the Indexes property of the C1.LiveLinq.Indexing.Search.IndexQuery<T> class, the result of an indexing search operation such as Index.Find and others, where it contains subindexes.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.Indexing.ScannerCollection<T>
    C1.LiveLinq.Indexing.IndexCollection<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ScannerCollection<T> Members
C1.LiveLinq.Indexing Namespace

Members
Properties Methods

The following tables list the members exposed by ScannerCollection<T>.

Public Properties

	Name	Description
	Count	Gets the number of indexes in the collection.

Top

Public Methods

	Name	Description
	Add	Overloaded. Creates a new index and adds it to the collection of indexes.

	Clear	Clears the collection of all indexes. All indexes are detached from the indexed collection and destroyed.
	Contains	Overloaded. Determines whether an index with the specified key selector exists in the collection.
	Find	Overloaded. Finds an index in the collection by its key selector.
	GetEnumerator	Returns an enumerator that iterates through the ScannerCollection<T>.
	Remove	Overloaded. Removes an index from the collection.

Top

See Also

Reference

ScannerCollection<T> Class
C1.LiveLinq.Indexing Namespace

Methods
For a list of all members of this type, see ScannerCollection<T> members.

Public Methods

	Name	Description
	Add	Overloaded. Creates a new index and adds it to the collection of indexes.
	Clear	Clears the collection of all indexes. All indexes are detached from the indexed collection and destroyed.
	Contains	Overloaded. Determines whether an index with the specified key selector exists in the collection.
	Find	Overloaded. Finds an index in the collection by its key selector.

	GetEnumerator	Returns an enumerator that iterates through the ScannerCollection<T>.
	Remove	Overloaded. Removes an index from the collection.

Top

See Also

Reference

ScannerCollection<T> Class
C1.LiveLinq.Indexing Namespace

Add Method
Creates a new index and adds it to the collection of indexes.

Overload List

Overload	Description
Add<TKey>(Expression<Func<T,TKey>>,Boolean,Boolean,IndexingAlgorithm,CultureInfo)	Creates a new index and adds it to the collection of indexes.
Add<TKey>(Expression<Func<T,TKey>>,Boolean,Boolean,CultureInfo)	Creates a new index and adds it to the collection of indexes.
Add<TKey>(Expression<Func<T,TKey>>,Boolean,Boolean)	Creates a

	new index and adds it to the collection of indexes.
Add<TKey>(Expression<Func<T,TKey>>,Boolean)	Creates a new index and adds it to the collection of indexes.
Add<TKey>(Expression<Func<T,TKey>>)	Creates a new index and adds it to the collection of indexes.
Add(LambdaExpression,Boolean,Boolean,IndexingAlgorithm,CultureInfo)	Creates a new index and adds it to the collection of indexes.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ScannerCollection<T> Class
ScannerCollection<T> Members

Add<TKey>(Expression<Func<T,TKey>>,Boolean,Boolean,IndexingAlgorithm,CultureInfo) Method
The type of the index key.

Key selector expression of the index, see KeySelector.

Specifies whether the key used in this index is a unique key for the indexed collection (default: **false**).

Specifies whether it is required that the index does not exist prior to this method call (default: **false**). If an index with this *keySelector* already exists, an exception is thrown if it is **true**, and this method call is ignored if it is **false**.

An IndexingAlgorithm used by the index. In the current version, only one algorithm is supported, RedBlackTree. Later versions may support other algorithms, such as bitmap or hash indexes.

Locale information used to compare strings in the index (default: CultureInfo.CurrentCulture).

Creates a new index and adds it to the collection of indexes.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function Add(Of TKey)( _
   ByVal keySelector As Expression(Of Func(Of T,TKey)), _
   ByVal keyIsUnique As Boolean, _
   ByVal onlyOnce As Boolean, _
   ByVal algorithm As IndexingAlgorithm, _
   ByVal locale As CultureInfo _
) As IIndexScanner(Of T,TKey)
```

C#

```
public IIndexScanner<T,TKey> Add<TKey>(
   Expression<Func<T,TKey>> keySelector,
   bool keyIsUnique,
   bool onlyOnce,
```

```
    IndexingAlgorithm algorithm,
    CultureInfo locale
)
```

Parameters

keySelector

Key selector expression of the index, see KeySelector.

keyIsUnique

Specifies whether the key used in this index is a unique key for the indexed collection (default: **false**).

onlyOnce

Specifies whether it is required that the index does not exist prior to this method call (default: **false**). If an index with this *keySelector* already exists, an exception is thrown if it is **true**, and this method call is ignored if it is **false**.

algorithm

An IndexingAlgorithm used by the index. In the current version, only one algorithm is supported, RedBlackTree. Later versions may support other algorithms, such as bitmap or hash indexes.

locale

Locale information used to compare strings in the index (default: CultureInfo.CurrentCulture).

Type Parameters

TKey

The type of the index key.

Return Value

The new index added to the collection of indexes.

Remarks

A unique index occupies less memory and performs better than a non-unique index (although the difference isn't dramatic). Therefore, for unique keys, it's recommended to specify the corresponding index as unique.

But do that only if you are sure that the key is indeed unique, as it imposes a uniqueness constraint on the indexed collection. An attempt to modify the indexed collection violating the uniqueness throws an System.InvalidOperationException.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ScannerCollection<T> Class
ScannerCollection<T> Members
Overload List

Add<TKey>(Expression<Func<T,TKey>>,Boolean,Boolean,CultureInfo) Method
The type of the index key.

Key selector expression of the index, see KeySelector.

Specifies whether the key used in this index is a unique key for the indexed collection (default: **false**).

Specifies whether it is required that the index does not exist prior to this method call (default: **false**). If an index with this *keySelector* already exists, an exception is thrown if it is **true**, and this method call is ignored if it is **false**.

Locale information used to compare strings in the index (default: CultureInfo.CurrentCulture).

Creates a new index and adds it to the collection of indexes.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function Add(Of TKey)( _
   ByVal keySelector As Expression(Of Func(Of T,TKey)), _
   ByVal keyIsUnique As Boolean, _
   ByVal onlyOnce As Boolean, _
```

```
   ByVal locale As CultureInfo _
) As IIndexScanner(Of T,TKey)
```

C#

```
public IIndexScanner<T,TKey> Add<TKey>(
   Expression<Func<T,TKey>> keySelector,
   bool keyIsUnique,
   bool onlyOnce,
   CultureInfo locale
)
```

Parameters

keySelector

Key selector expression of the index, see KeySelector.

keyIsUnique

Specifies whether the key used in this index is a unique key for the indexed collection (default: **false**).

onlyOnce

Specifies whether it is required that the index does not exist prior to this method call (default: **false**). If an index with this *keySelector* already exists, an exception is thrown if it is **true**, and this method call is ignored if it is **false**.

locale

Locale information used to compare strings in the index (default: CultureInfo.CurrentCulture).

Type Parameters

TKey

The type of the index key.

Return Value

The new index added to the collection of indexes.

Remarks

A unique index occupies less memory and performs better than a non-unique index (although the difference isn't dramatic). Therefore, for unique keys, it's recommended to specify the corresponding index as unique.

But do that only if you are sure that the key is indeed unique, as it imposes a uniqueness constraint on the indexed collection. An attempt to modify the indexed collection violating the uniqueness throws an System.InvalidOperationException.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ScannerCollection<T> Class
ScannerCollection<T> Members
Overload List

Add<TKey>(Expression<Func<T,TKey>>,Boolean,Boolean) Method

The type of the index key.

Key selector expression of the index, see KeySelector.

Specifies whether the key used in this index is a unique key for the indexed collection (default: **false**).

Specifies whether it is required that the index does not exist prior to this method call (default: **false**). If an index with this *keySelector* already exists, an exception is thrown if it is **true**, and this method call is ignored if it is **false**.

Creates a new index and adds it to the collection of indexes.

Syntax

Visual Basic (Declaration)
`Public Overloads Function Add(Of TKey)( _`

```
   ByVal keySelector As Expression(Of Func(Of T,TKey)), _
   ByVal keyIsUnique As Boolean, _
   ByVal onlyOnce As Boolean _
) As IIndexScanner(Of T,TKey)
```

C#

```
public IIndexScanner<T,TKey> Add<TKey>(
   Expression<Func<T,TKey>> keySelector,
   bool keyIsUnique,
   bool onlyOnce
)
```

Parameters

keySelector

Key selector expression of the index, see KeySelector.

keyIsUnique

Specifies whether the key used in this index is a unique key for the indexed collection (default: **false**).

onlyOnce

Specifies whether it is required that the index does not exist prior to this method call (default: **false**). If an index with this *keySelector* already exists, an exception is thrown if it is **true**, and this method call is ignored if it is **false**.

Type Parameters

TKey

The type of the index key.

Return Value

The new index added to the collection of indexes.

Remarks

A unique index occupies less memory and performs better than a non-unique index (although the difference isn't dramatic). Therefore, for unique keys, it's recommended to specify the corresponding index as unique.

But do that only if you are sure that the key is indeed unique, as it imposes a uniqueness constraint on the indexed collection. An attempt to modify the indexed collection violating the uniqueness throws an System.InvalidOperationException.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ScannerCollection<T> Class
ScannerCollection<T> Members
Overload List

Add<TKey>(Expression<Func<T,TKey>>,Boolean) Method
The type of the index key.

Key selector expression of the index, see KeySelector.

Specifies whether the key used in this index is a unique key for the indexed collection (default: **false**).

Creates a new index and adds it to the collection of indexes.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function Add(Of TKey)( _
   ByVal keySelector As Expression(Of Func(Of T,TKey)), _
   ByVal keyIsUnique As Boolean _
) As IIndexScanner(Of T,TKey)
```

C#

```
public IIndexScanner<T,TKey> Add<TKey>(
   Expression<Func<T,TKey>> keySelector,
   bool keyIsUnique
)
```

Parameters

keySelector

Key selector expression of the index, see KeySelector.

keyIsUnique

Specifies whether the key used in this index is a unique key for the indexed collection (default: **false**).

Type Parameters

TKey

The type of the index key.

Return Value

The new index added to the collection of indexes.

Remarks

A unique index occupies less memory and performs better than a non-unique index (although the difference isn't dramatic). Therefore, for unique keys, it's recommended to specify the corresponding index as unique.

But do that only if you are sure that the key is indeed unique, as it imposes a uniqueness constraint on the indexed collection. An attempt to modify the indexed collection violating the uniqueness throws an System.InvalidOperationException.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ScannerCollection<T> Class
ScannerCollection<T> Members
Overload List

Add<TKey>(Expression<Func<T,TKey>>) Method
The type of the index key.

Key selector expression of the index, see KeySelector.

Creates a new index and adds it to the collection of indexes.

Syntax

Visual Basic (Declaration)

`Public Overloads Function Add(Of TKey)( _`
`  ByVal keySelector As Expression(Of Func(Of T,TKey)) _`
`) As IIndexScanner(Of T,TKey)`

C#

`public IIndexScanner<T,TKey> Add<TKey>(`
`  Expression<Func<T,TKey>> keySelector`
`)`

Parameters

keySelector

Key selector expression of the index, see KeySelector.

Type Parameters

TKey

The type of the index key.

Return Value

The new index added to the collection of indexes.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ScannerCollection<T> Class
ScannerCollection<T> Members
Overload List

Add(LambdaExpression,Boolean,Boolean,IndexingAlgorithm,CultureInfo) Method

Key selector expression of the index, see KeySelector.

Specifies whether the key used in this index is a unique key for the indexed collection (default: **false**).

Specifies whether it is required that the index does not exist prior to this method call. If an index with this *keySelector* already exists, an exception is thrown if it is **true**, and this method call is ignored if it is **false**.

An IndexingAlgorithm used by the index. In the current version, only one algorithm is supported, RedBlackTree. Later versions may support other algorithms, such as bitmap or hash indexes.

Locale information used to compare strings in the index (default: CultureInfo.CurrentCulture).

Creates a new index and adds it to the collection of indexes.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function Add( _
   ByVal keySelector As LambdaExpression, _
   ByVal keyIsUnique As Boolean, _
   ByVal onlyOnce As Boolean, _
   ByVal algorithm As IndexingAlgorithm, _
   ByVal locale As CultureInfo _
) As IIndexScanner(Of T)
```

C#

```
public IIndexScanner<T> Add(
   LambdaExpression keySelector,
   bool keyIsUnique,
   bool onlyOnce,
   IndexingAlgorithm algorithm,
   CultureInfo locale
)
```

Parameters

keySelector

Key selector expression of the index, see KeySelector.

keyIsUnique

Specifies whether the key used in this index is a unique key for the indexed collection (default: **false**).

onlyOnce

Specifies whether it is required that the index does not exist prior to this method call. If an index with this *keySelector* already exists, an exception is thrown if it is **true**, and this method call is ignored if it is **false**.

algorithm

An IndexingAlgorithm used by the index. In the current version, only one algorithm is supported, RedBlackTree. Later versions may support other algorithms, such as bitmap or hash indexes.

locale

Locale information used to compare strings in the index (default: CultureInfo.CurrentCulture).

Return Value

The new index added to the collection of indexes.

Remarks

A unique index occupies less memory and performs better than a non-unique index (although the difference isn't dramatic). Therefore, for unique keys, it's recommended to specify the corresponding index as unique.

But do that only if you are sure that the key is indeed unique, as it imposes a uniqueness constraint on the indexed collection. An attempt to modify the indexed collection violating the uniqueness throws an System.InvalidOperationException.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server

2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ScannerCollection<T> Class
ScannerCollection<T> Members
Overload List

Clear Method

Clears the collection of all indexes. All indexes are detached from the indexed collection and destroyed.

Syntax

Visual Basic (Declaration)
`Public MustOverride Sub Clear()`
C#
`public abstract void Clear()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ScannerCollection<T> Class
ScannerCollection<T> Members

Contains Method

Determines whether an index with the specified key selector exists in the collection.

Overload List

Overload	Description
Contains(LambdaExpression)	Determines whether an index with the specified key selector exists in the collection.
Contains<TKey>(Expression<Func<T,TKey>>)	Determines whether an index with the specified key selector exists in the collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ScannerCollection<T> Class
ScannerCollection<T> Members

Contains(LambdaExpression) Method
Key selector expression of an index, see KeySelector.

Determines whether an index with the specified key selector exists in the collection.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function Contains( _
   ByVal keySelector As LambdaExpression _
) As Boolean
```

C#

```
public bool Contains(
   LambdaExpression keySelector
)
```

Parameters

keySelector

Key selector expression of an index, see KeySelector.

Return Value

true if an index with the specified key selector is found in the collection; otherwise, **false**.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ScannerCollection<T> Class
ScannerCollection<T> Members
Overload List

Contains<TKey>(Expression<Func<T,TKey>>) Method
The type of the index key.

Key selector expression of an index, see KeySelector.

Determines whether an index with the specified key selector exists in the collection.

Syntax

Visual Basic (Declaration)

`Public Overloads Function Contains(Of TKey)( _`
`  ByVal keySelector As Expression(Of Func(Of T,TKey)) _`
`) As Boolean`

C#

`public bool Contains<TKey>(`
`  Expression<Func<T,TKey>> keySelector`
`)`

Parameters

keySelector

Key selector expression of an index, see KeySelector.

Type Parameters

TKey

The type of the index key.

Return Value

true if an index with the specified key selector is found in the collection; otherwise, **false**.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ScannerCollection<T> Class
ScannerCollection<T> Members
Overload List

Find Method
Finds an index in the collection by its key selector.

Overload List

Overload	Description
Find(LambdaExpression)	Finds an index in the collection by its key selector.
Find<TKey>(Expression<Func<T,TKey>>)	Finds an index in the collection by its key selector.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ScannerCollection<T> Class
ScannerCollection<T> Members

Find(LambdaExpression) Method
Key selector expression of an index, see KeySelector.

Finds an index in the collection by its key selector.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function Find( _
   ByVal keySelector As LambdaExpression _
) As IIndexScanner(Of T)
```

C#

```
public IIndexScanner<T> Find(
   LambdaExpression keySelector
)
```

Parameters

keySelector

Key selector expression of an index, see KeySelector.

Return Value

An index with the given key selector, if it is found; otherwise, **null**.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ScannerCollection<T> Class
ScannerCollection<T> Members
Overload List

The type of the index key.

Key selector expression of an index, see KeySelector.

Finds an index in the collection by its key selector.

Syntax

Visual Basic (Declaration)

`Public Overloads Function Find(Of TKey)( _`
`  ByVal keySelector As Expression(Of Func(Of T,TKey)) _`
`) As IIndexScanner(Of T,TKey)`

C#

`public IIndexScanner<T,TKey> Find<TKey>(`
`  Expression<Func<T,TKey>> keySelector`
`)`

Parameters

keySelector

Key selector expression of an index, see KeySelector.

Type Parameters

TKey

The type of the index key.

Return Value

An index with the given key selector, if it is found; otherwise, **null**.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ScannerCollection<T> Class
ScannerCollection<T> Members
Overload List

GetEnumerator Method
Returns an enumerator that iterates through the ScannerCollection<T>.

Syntax

Visual Basic (Declaration)
`Public MustOverride Function GetEnumerator() As IEnumerator(Of IIndexScanner(Of T))`
C#
`public abstract IEnumerator<IIndexScanner<T>> GetEnumerator()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ScannerCollection<T> Class
ScannerCollection<T> Members

Remove Method
Removes an index from the collection.

Overload List

Overload	Description
Remove(LambdaExpression)	Removes an index from the collection.
Remove<TKey>(Expression<Func<T,TKey>>)	Removes an index from the collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ScannerCollection<T> Class
ScannerCollection<T> Members

Remove(LambdaExpression) Method
Key selector expression of an index, see KeySelector.

Removes an index from the collection.

Syntax

Visual Basic (Declaration)

```
Public Overloads MustOverride Function Remove( _
   ByVal keySelector As LambdaExpression _
) As Boolean
```

C#

```
public abstract bool Remove(
   LambdaExpression keySelector
)
```

Parameters

keySelector

Key selector expression of an index, see KeySelector.

Return Value

true if an index has been removed; **false** if there is no index with the given key selector in the collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ScannerCollection<T> Class
ScannerCollection<T> Members
Overload List

Remove<TKey>(Expression<Func<T,TKey>>) Method
The type of the index key.

Key selector expression of an index, see KeySelector.

Removes an index from the collection.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function Remove(Of TKey)( _
   ByVal keySelector As Expression(Of Func(Of T,TKey)) _
) As Boolean
```

C#

```
public bool Remove<TKey>(
   Expression<Func<T,TKey>> keySelector
)
```

Parameters

keySelector

Key selector expression of an index, see KeySelector.

Type Parameters

TKey

The type of the index key.

Return Value

true if an index has been removed; **false** if there is no index with the given key selector in the collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ScannerCollection<T> Class
ScannerCollection<T> Members
Overload List

Properties
For a list of all members of this type, see ScannerCollection<T> members.

Public Properties

	Name	Description
	Count	Gets the number of indexes in the collection.

Top

See Also

Reference

ScannerCollection<T> Class
C1.LiveLinq.Indexing Namespace

Count Property
Gets the number of indexes in the collection.

Syntax

Visual Basic (Declaration)
`Public MustOverride ReadOnly Property Count As Integer`

C#

```
public abstract int Count {get;}
```

Property Value

The number of indexes in the collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

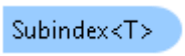
Reference

ScannerCollection<T> Class
ScannerCollection<T> Members

Subindex<T>
The type of the elements of the collection to index.

Base class for the Subindex<T,TKey> class.

Object Model



Syntax

Visual Basic (Declaration)

```
Public MustInherit Class Subindex(Of T)
   Inherits IndexDefinition(Of T)
```

C#

```
public abstract class Subindex<T> : IndexDefinition<T>
```

Type Parameters

T

The type of the elements of the collection to index.

Remarks

You don't typically use the Subindex<T> class directly. It provides functionality of the Subindex<T,TKey> class that does not depend on the index key type. The base class Subindex<T> is needed only if the index key type is not known, usually in general-purpose code intended for reuse with different key types.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.Indexing.IndexDefinition<T>
    C1.LiveLinq.Indexing.Subindex<T>
      C1.LiveLinq.Indexing.Subindex<T,TKey>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

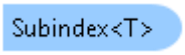
Reference

Subindex<T> Members
C1.LiveLinq.Indexing Namespace

Overview
The type of the elements of the collection to index.

Base class for the Subindex<T,TKey> class.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit Class Subindex(Of T)`

```
Inherits IndexDefinition(Of T)
```

C#

```
public abstract class Subindex<T> : IndexDefinition<T>
```

Type Parameters

T

The type of the elements of the collection to index.

Remarks

You don't typically use the Subindex<T> class directly. It provides functionality of the Subindex<T,TKey> class that does not depend on the index key type. The base class Subindex<T> is needed only if the index key type is not known, usually in general-purpose code intended for reuse with different key types.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.Indexing.IndexDefinition<T>
    C1.LiveLinq.Indexing.Subindex<T>
      C1.LiveLinq.Indexing.Subindex<T,TKey>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Subindex<T> Members
C1.LiveLinq.Indexing Namespace

Members
Properties

The following tables list the members exposed by Subindex<T>.

Public Properties

	Name	Description
	Algorithm	Gets the indexing algorithm used by the index. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	KeyIsUnique	Gets a value that indicates whether the key used in this index is a unique key for the collection. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	KeySelector	Gets the expression used to obtain key value from an element of the indexed collection. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	KeyType	Gets the type of the index key. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	Locale	Gets the locale information used to compare strings in the index. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	Parent	Parent of this subindex definition in the index/subindexes hierarchy.
	Root	Gets the root index in an index/subindex hierarchy. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	Subindexes	Gets the collection of subindexes added to this index. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)

Top

See Also

Reference

Subindex<T> Class
C1.LiveLinq.Indexing Namespace

Properties
For a list of all members of this type, see Subindex<T> members.

Public Properties

	Name	Description
	Algorithm	Gets the indexing algorithm used by the index. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	KeyIsUnique	Gets a value that indicates whether the key used in this index is a unique key for the collection. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	KeySelector	Gets the expression used to obtain key value from an element of the indexed collection. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	KeyType	Gets the type of the index key. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	Locale	Gets the locale information used to compare strings in the index. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	Parent	Parent of this subindex definition in the index/subindexes hierarchy.
	Root	Gets the root index in an index/subindex hierarchy. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	Subindexes	Gets the collection of subindexes added to this index. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)

Top

See Also

Reference

Subindex<T> Class
C1.LiveLinq.Indexing Namespace

Parent Property
Parent of this subindex definition in the index/subindexes hierarchy.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Property Parent As IndexDefinition(Of T)
```

C#

```
public IndexDefinition<T> Parent {get;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

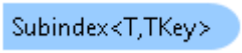
Subindex<T> Class
Subindex<T> Members

Subindex<T,TKey>
The type of the elements of the collection to index.

The type of the index key.

Defines a subindex, an index definition subordinate to another index definition, its parent.

Object Model



Syntax

Visual Basic (Declaration)

```
Public MustInherit Class Subindex
    (Of T,TKey)
   Inherits Subindex(Of T)
```

C#

```
public abstract class Subindex<T,TKey> : Subindex<T>
```

Type Parameters

T

The type of the elements of the collection to index.

TKey

The type of the index key.

Remarks

An index (Index<T,TKey>) can have subindexes. Subindexes are optional, not required for any indexing tasks, but can provide additional optimization and help minimize memory requirements when a collection is indexed by multi-level (multi-field) keys.

Suppose we want to index a Customers table by two fields, (City, Rating), perhaps for speeding up queries like `from c in Customers where c.City == "London" && c.Rating == 1 select c` We can do it by defining an index with key selector `c => new { c.City, c.Rating }`, that will index the table by two fields, creating a multi-field index. Such index will suffice for optimizing the query above, but it will not optimize, for example, the following query: `from c in Customers where c.City == "London" && c.Rating > 2 select c` Also, multi-field indexes occupy more memory than necessary because they have to store repeated field values.

Subindexes provide a better alternative for optimizing multi-field searches. In the example above, we can define an index by City and create a subindex of that index, by Rating. Using subindexes becomes even more effective when, as it is often happens, we also need queries to search by additional fields, like, for example, if we need to search by ContactTitle inside a city in addition to the search by Rating inside a city: `from c in Customers where c.City == "London" && c.ContactTitle == "Owner" select c` All we have to do now is to add a second subindex to the index by City, a subindex by ContactTitle.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.Indexing.IndexDefinition<T>
    C1.LiveLinq.Indexing.Subindex<T>
      C1.LiveLinq.Indexing.Subindex<T,TKey>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

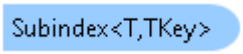
Subindex<T,TKey> Members
C1.LiveLinq.Indexing Namespace

Overview
The type of the elements of the collection to index.

The type of the index key.

Defines a subindex, an index definition subordinate to another index definition, its parent.

Object Model



Syntax

Visual Basic (Declaration)

`Public MustInherit Class Subindex`
`  (Of T,TKey)`
`  Inherits Subindex(Of T)`

C#
`public abstract class Subindex<T,TKey> : Subindex<T>`

Type Parameters

T

The type of the elements of the collection to index.

TKey

The type of the index key.

Remarks

An index (Index<T,TKey>) can have subindexes. Subindexes are optional, not required for any indexing tasks, but can provide additional optimization and help minimize memory requirements when a collection is indexed by multi-level (multi-field) keys.

Suppose we want to index a Customers table by two fields, (City, Rating), perhaps for speeding up queries like `from c in Customers where c.City == "London" && c.Rating == 1 select c` We can do it by defining an index with key selector `c => new { c.City, c.Rating }`, that will index the table by two fields, creating a multi-field index. Such index will suffice for optimizing the query above, but it will not optimize, for example, the following query: `from c in Customers where c.City == "London" && c.Rating > 2 select c` Also, multi-field indexes occupy more memory than necessary because they have to store repeated field values.

Subindexes provide a better alternative for optimizing multi-field searches. In the example above, we can define an index by City and create a subindex of that index, by Rating. Using subindexes becomes even more effective when, as it is often happens, we also need queries to search by additional fields, like, for example, if we need to search by ContactTitle inside a city in addition to the search by Rating inside a city: `from c in Customers where c.City == "London" && c.ContactTitle == "Owner" select c` All we have to do now is to add a second subindex to the index by City, a subindex by ContactTitle.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.Indexing.IndexDefinition<T>
    C1.LiveLinq.Indexing.Subindex<T>
      C1.LiveLinq.Indexing.Subindex<T,TKey>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

Subindex<T,TKey> Members
C1.LiveLinq.Indexing Namespace

Members
Properties

The following tables list the members exposed by Subindex<T,TKey>.

Public Properties

	Name	Description
	Algorithm	Gets the indexing algorithm used by the index. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	KeyIsUnique	Gets a value that indicates whether the key used in this index is a unique key for the collection. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	KeySelector	Gets the expression used to obtain key value from an element of the indexed collection.
	KeyType	Gets the type of the index key. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	Locale	Gets the locale information used to compare strings in the index. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	Parent	Parent of this subindex definition in the index/subindexes hierarchy. (Inherited from C1.LiveLinq.Indexing.Subindex<T>)
	Root	Gets the root index in an index/subindex hierarchy. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	Subindexes	Gets the collection of subindexes added to this index. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)

Top

See Also

Reference

Subindex<T,TKey> Class
C1.LiveLinq.Indexing Namespace

Properties

For a list of all members of this type, see Subindex<T,TKey> members.

Public Properties

	Name	Description
	Algorithm	Gets the indexing algorithm used by the index. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	KeyIsUnique	Gets a value that indicates whether the key used in this index is a unique key for the collection. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	KeySelector	Gets the expression used to obtain key value from an element of the indexed collection.
	KeyType	Gets the type of the index key. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	Locale	Gets the locale information used to compare strings in the index. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	Parent	Parent of this subindex definition in the index/subindexes hierarchy. (Inherited from C1.LiveLinq.Indexing.Subindex<T>)
	Root	Gets the root index in an index/subindex hierarchy. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)
	Subindexes	Gets the collection of subindexes added to this index. (Inherited from C1.LiveLinq.Indexing.IndexDefinition<T>)

Top

See Also

Reference

Subindex<T,TKey> Class
C1.LiveLinq.Indexing Namespace

KeySelector Property
Gets the expression used to obtain key value from an element of the indexed collection.

Syntax

Visual Basic (Declaration)
`Public Shadows ReadOnly Property KeySelector As Expression(Of Func(Of T,TKey))`
C#
`public new Expression<Func<T,TKey>> KeySelector {get;}`

Property Value

An expression calculating the key value from an item (element of the collection). Typically, this is a field or a property in the item class, although more complex expressions can also be used.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

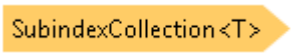
Reference

Subindex<T,TKey> Class
Subindex<T,TKey> Members

SubindexCollection<T>
The type of the elements of the collection to index.

Represents a collection of subindexes attached to an IndexDefinition<T>.

Object Model



Syntax

Visual Basic (Declaration)	
`Public NotInheritable Class SubindexCollection(Of T)`	
C#	
`public sealed class SubindexCollection<T>`	

Type Parameters

T

The type of the elements of the collection to index.

Remarks

A Subindex<T,TKey> is attached to its parent index definition, which is an Index<T,TKey> or another Subindex<T,TKey>. The subindexes collection is stored in the parent's IndexDefinition<T>.Subindexes property.

Subindexes are optional, not required for any indexing tasks, but can provide additional optimization and help minimize memory requirements when a collection is indexed by multi-level (multi-field) keys.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.Indexing.SubindexCollection<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SubindexCollection<T> Members
C1.LiveLinq.Indexing Namespace

Overview
The type of the elements of the collection to index.

Represents a collection of subindexes attached to an IndexDefinition<T>.

Object Model

Syntax

Visual Basic (Declaration)
`Public NotInheritable Class SubindexCollection(Of T)`
C#
`public sealed class SubindexCollection<T>`

Type Parameters

T

The type of the elements of the collection to index.

Remarks

A Subindex<T,TKey> is attached to its parent index definition, which is an Index<T,TKey> or another Subindex<T,TKey>. The subindexes collection is stored in the parent's IndexDefinition<T>.Subindexes property.

Subindexes are optional, not required for any indexing tasks, but can provide additional optimization and help minimize memory requirements when a collection is indexed by multi-level (multi-field) keys.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.Indexing.SubindexCollection<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SubindexCollection<T> Members
C1.LiveLinq.Indexing Namespace

Members
Properties Methods

The following tables list the members exposed by SubindexCollection<T>.

Public Properties

	Name	Description
	Count	Gets the number of subindexes in the collection.
	Item	Gets the subindex object at the specified ordinal position in the collection.

Top

Public Methods

	Name	Description
	Add	Overloaded. Creates a new subindex and attaches it to its parent's IndexDefinition<T>.Subindexes collection.
	Clear	Clears the collection of all subindexes. All subindexes are detached from the parent and destroyed.
	Contains	Overloaded. Determines whether a subindex with the specified key selector exists in the collection.
	Find	Overloaded. Finds a subindex in the collection by its key selector.
	GetEnumerator	Returns an enumerator that iterates through the SubindexCollection<T>.

	Remove	Overloaded. Removes a subindex from the collection.

Top

See Also

Reference

SubindexCollection<T> Class
C1.LiveLinq.Indexing Namespace

Methods
For a list of all members of this type, see SubindexCollection<T> members.

Public Methods

	Name	Description
	Add	Overloaded. Creates a new subindex and attaches it to its parent's IndexDefinition<T>.Subindexes collection.
	Clear	Clears the collection of all subindexes. All subindexes are detached from the parent and destroyed.
	Contains	Overloaded. Determines whether a subindex with the specified key selector exists in the collection.
	Find	Overloaded. Finds a subindex in the collection by its key selector.
	GetEnumerator	Returns an enumerator that iterates through the SubindexCollection<T>.
	Remove	Overloaded. Removes a subindex from the collection.

Top

See Also

Reference

SubindexCollection<T> Class
C1.LiveLinq.Indexing Namespace

Add Method

Creates a new subindex and attaches it to its parent's IndexDefinition<T>.Subindexes collection.

Overload List

Overload	Description
Add<TKey>(Expression<Func<T,TKey>>,Boolean,Boolean,IndexingAlgorithm,CultureInfo)	Creates a new subindex and attaches it to its parent's IndexDefinition<T>.Subindexes collection.
Add<TKey>(Expression<Func<T,TKey>>,Boolean,Boolean,CultureInfo)	Creates a new subindex and attaches it to its parent's IndexDefinition<T>.Subindexes collection.
Add<TKey>(Expression<Func<T,TKey>>,Boolean,Boolean)	Creates a new subindex and attaches it to its parent's IndexDefinition<T>.Subindexes collection.
Add<TKey>(Expression<Func<T,TKey>>,Boolean)	Creates a new subindex and attaches it to its parent's IndexDefinition<T>.Subindexes collection.
Add<TKey>(Expression<Func<T,TKey>>)	Creates a new subindex and attaches it to its parent's IndexDefinition<T>.Subi

	ndexes collection.
Add(LambdaExpression,Boolean,Boolean,IndexingAlgorithm,CultureInfo)	Creates a new subindex and attaches it to its parent's IndexDefinition<T>.Subindexes collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SubindexCollection<T> Class
SubindexCollection<T> Members

Add<TKey>(Expression<Func<T,TKey>>,Boolean,Boolean,IndexingAlgorithm,CultureInfo) Method
The type of the subindex key.

Key selector expression of the subindex, see KeySelector.

Specifies whether the key used in this subindex is unique for any given value of the parent key (default: **false**).

Specifies whether it is required that the subindex does not exist prior to this method call (default: **false**). If a subindex with this *keySelector* already exists, an exception is thrown if it is **true**, and this method call is ignored if it is **false**.

An IndexingAlgorithm used by the subindex. In the current version, only one algorithm is supported, RedBlackTree. Later versions may support other algorithms, such as bitmap or hash indexing.

Locale information used to compare strings in the subindex (default: CultureInfo.CurrentCulture).

Creates a new subindex and attaches it to its parent's IndexDefinition<T>.Subindexes collection.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function Add(Of TKey)( _
   ByVal keySelector As Expression(Of Func(Of T,TKey)), _
   ByVal keyIsUnique As Boolean, _
   ByVal onlyOnce As Boolean, _
   ByVal algorithm As IndexingAlgorithm, _
   ByVal locale As CultureInfo _
) As Subindex(Of T,TKey)
```

C#

```
public Subindex<T,TKey> Add<TKey>(
   Expression<Func<T,TKey>> keySelector,
   bool keyIsUnique,
   bool onlyOnce,
   IndexingAlgorithm algorithm,
   CultureInfo locale
)
```

Parameters

keySelector

Key selector expression of the subindex, see KeySelector.

keyIsUnique

Specifies whether the key used in this subindex is unique for any given value of the parent key (default: **false**).

onlyOnce

Specifies whether it is required that the subindex does not exist prior to this method call (default: **false**). If a subindex with this *keySelector* already exists, an exception is thrown if it is **true**, and this method call is ignored if it is **false**.

algorithm

An IndexingAlgorithm used by the subindex. In the current version, only one algorithm is supported, RedBlackTree. Later versions may support other algorithms, such as bitmap or hash indexing.

locale

Locale information used to compare strings in the subindex (default: CultureInfo.CurrentCulture).

Type Parameters

TKey

The type of the subindex key.

Return Value

The new subindex added to its parent's IndexDefinition<T>.Subindexes collection.

Remarks

A unique index occupies less memory and performs better than a non-unique index (although the difference isn't dramatic). Therefore, for unique keys, it's recommended to specify the corresponding index as unique.

But do that only if you are sure that the key is indeed unique, as it imposes a uniqueness constraint on the indexed collection. An attempt to modify the indexed collection violating the uniqueness throws an System.InvalidOperationException.

For a subindex, uniqueness means that any given pair of parent key and subindex key values uniquely determines an item in the indexed collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SubindexCollection<T> Class
SubindexCollection<T> Members
Overload List

The type of the subindex key.

Key selector expression of the subindex, see KeySelector.

Specifies whether the key used in this subindex is unique for any given value of the parent key (default: **false**).

Specifies whether it is required that the subindex does not exist prior to this method call (default: **false**). If a subindex with this *keySelector* already exists, an exception is thrown if it is **true**, and this method call is ignored if it is **false**.

Locale information used to compare strings in the subindex (default: CultureInfo.CurrentCulture).

Creates a new subindex and attaches it to its parent's IndexDefinition<T>.Subindexes collection.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function Add(Of TKey)( _
   ByVal keySelector As Expression(Of Func(Of T,TKey)), _
   ByVal keyIsUnique As Boolean, _
   ByVal onlyOnce As Boolean, _
   ByVal locale As CultureInfo _
) As Subindex(Of T,TKey)
```

C#

```
public Subindex<T,TKey> Add<TKey>(
   Expression<Func<T,TKey>> keySelector,
   bool keyIsUnique,
   bool onlyOnce,
   CultureInfo locale
)
```

Parameters

keySelector

Key selector expression of the subindex, see KeySelector.

keyIsUnique

Specifies whether the key used in this subindex is unique for any given value of the parent key (default: **false**).

onlyOnce

Specifies whether it is required that the subindex does not exist prior to this method call (default: **false**). If a subindex with this *keySelector* already exists, an exception is thrown if it is **true**, and this method call is ignored if it is **false**.

locale

Locale information used to compare strings in the subindex (default: CultureInfo.CurrentCulture).

Type Parameters

TKey

The type of the subindex key.

Return Value

The new subindex added to its parent's IndexDefinition<T>.Subindexes collection.

Remarks

A unique index occupies less memory and performs better than a non-unique index (although the difference isn't dramatic). Therefore, for unique keys, it's recommended to specify the corresponding index as unique.

But do that only if you are sure that the key is indeed unique, as it imposes a uniqueness constraint on the indexed collection. An attempt to modify the indexed collection violating the uniqueness throws an System.InvalidOperationException.

For a subindex, uniqueness means that any given pair of parent key and subindex key values uniquely determines an item in the indexed collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server

2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SubindexCollection<T> Class
SubindexCollection<T> Members
Overload List

Add<TKey>(Expression<Func<T,TKey>>,Boolean,Boolean) Method
The type of the subindex key.

Key selector expression of the subindex, see KeySelector.

Specifies whether the key used in this subindex is unique for any given value of the parent key (default: **false**).

Specifies whether it is required that the subindex does not exist prior to this method call (default: **false**). If a subindex with this *keySelector* already exists, an exception is thrown if it is **true**, and this method call is ignored if it is **false**.

Creates a new subindex and attaches it to its parent's IndexDefinition<T>.Subindexes collection.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function Add(Of TKey)( _
   ByVal keySelector As Expression(Of Func(Of T,TKey)), _
   ByVal keyIsUnique As Boolean, _
   ByVal onlyOnce As Boolean _
) As Subindex(Of T,TKey)
```

C#

```
public Subindex<T,TKey> Add<TKey>(
   Expression<Func<T,TKey>> keySelector,
   bool keyIsUnique,
   bool onlyOnce
)
```

Parameters

keySelector

Key selector expression of the subindex, see KeySelector.

keyIsUnique

Specifies whether the key used in this subindex is unique for any given value of the parent key (default: **false**).

onlyOnce

Specifies whether it is required that the subindex does not exist prior to this method call (default: **false**). If a subindex with this *keySelector* already exists, an exception is thrown if it is **true**, and this method call is ignored if it is **false**.

Type Parameters

TKey

The type of the subindex key.

Return Value

The new subindex added to its parent's IndexDefinition<T>.Subindexes collection.

Remarks

A unique index occupies less memory and performs better than a non-unique index (although the difference isn't dramatic). Therefore, for unique keys, it's recommended to specify the corresponding index as unique.

But do that only if you are sure that the key is indeed unique, as it imposes a uniqueness constraint on the indexed collection. An attempt to modify the indexed collection violating the uniqueness throws an System.InvalidOperationException.

For a subindex, uniqueness means that any given pair of parent key and subindex key values uniquely determines an item in the indexed collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SubindexCollection<T> Class
SubindexCollection<T> Members
Overload List

Add<TKey>(Expression<Func<T,TKey>>,Boolean) Method
The type of the subindex key.

Key selector expression of the subindex, see KeySelector.

Specifies whether the key used in this subindex is unique for any given value of the parent key (default: **false**).

Creates a new subindex and attaches it to its parent's IndexDefinition<T>.Subindexes collection.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function Add(Of TKey)( _
   ByVal keySelector As Expression(Of Func(Of T,TKey)), _
   ByVal keyIsUnique As Boolean _
) As Subindex(Of T,TKey)
```

C#

```
public Subindex<T,TKey> Add<TKey>(
   Expression<Func<T,TKey>> keySelector,
   bool keyIsUnique
)
```

Parameters

keySelector

Key selector expression of the subindex, see KeySelector.

keyIsUnique

Specifies whether the key used in this subindex is unique for any given value of the parent key (default: **false**).

Type Parameters

TKey

The type of the subindex key.

Return Value

The new subindex added to its parent's IndexDefinition<T>.Subindexes collection.

Remarks

A unique index occupies less memory and performs better than a non-unique index (although the difference isn't dramatic). Therefore, for unique keys, it's recommended to specify the corresponding index as unique.

But do that only if you are sure that the key is indeed unique, as it imposes a uniqueness constraint on the indexed collection. An attempt to modify the indexed collection violating the uniqueness throws an System.InvalidOperationException.

For a subindex, uniqueness means that any given pair of parent key and subindex key values uniquely determines an item in the indexed collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SubindexCollection<T> Class
SubindexCollection<T> Members
Overload List

Add<TKey>(Expression<Func<T,TKey>>) Method
The type of the subindex key.

Key selector expression of the subindex, see KeySelector.

Creates a new subindex and attaches it to its parent's IndexDefinition<T>.Subindexes collection.

Syntax

Visual Basic (Declaration)	

```
Public Overloads Function Add(Of TKey)( _
   ByVal keySelector As Expression(Of Func(Of T,TKey)) _
) As Subindex(Of T,TKey)
```

C#

```
public Subindex<T,TKey> Add<TKey>(
   Expression<Func<T,TKey>> keySelector
)
```

Parameters

keySelector

Key selector expression of the subindex, see KeySelector.

Type Parameters

TKey

The type of the subindex key.

Return Value

The new subindex added to its parent's IndexDefinition<T>.Subindexes collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SubindexCollection<T> Class
SubindexCollection<T> Members
Overload List

Add(LambdaExpression,Boolean,Boolean,IndexingAlgorithm,CultureInfo) Method
Key selector expression of the subindex, see KeySelector.

Specifies whether the key used in this subindex is unique for any given value of the parent key (default: **false**).

Specifies whether it is required that the subindex does not exist prior to this method call. If a subindex with this *keySelector* already exists, an exception is thrown if it is **true**, and this method call is ignored if it is **false**.

An IndexingAlgorithm used by the subindex. In the current version, only one algorithm is supported, RedBlackTree. Later versions may support other algorithms, such as bitmap or hash indexing.

Locale information used to compare strings in the subindex (default: CultureInfo.CurrentCulture).

Creates a new subindex and attaches it to its parent's IndexDefinition<T>.Subindexes collection.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function Add( _
   ByVal keySelector As LambdaExpression, _
   ByVal keyIsUnique As Boolean, _
   ByVal onlyOnce As Boolean, _
   ByVal algorithm As IndexingAlgorithm, _
   ByVal locale As CultureInfo _
) As Subindex(Of T)
```

C#

```
public Subindex<T> Add(
   LambdaExpression keySelector,
   bool keyIsUnique,
   bool onlyOnce,
   IndexingAlgorithm algorithm,
   CultureInfo locale
)
```

Parameters

keySelector

Key selector expression of the subindex, see KeySelector.

keyIsUnique

Specifies whether the key used in this subindex is unique for any given value of the parent key (default: **false**).

onlyOnce

Specifies whether it is required that the subindex does not exist prior to this method call. If a subindex with this *keySelector* already exists, an exception is thrown if it is **true**, and this method call is ignored if it is **false**.

algorithm

An IndexingAlgorithm used by the subindex. In the current version, only one algorithm is supported, RedBlackTree. Later versions may support other algorithms, such as bitmap or hash indexing.

locale

Locale information used to compare strings in the subindex (default: CultureInfo.CurrentCulture).

Return Value

The new subindex added to its parent's IndexDefinition<T>.Subindexes collection.

Remarks

A unique index occupies less memory and performs better than a non-unique index (although the difference isn't dramatic). Therefore, for unique keys, it's recommended to specify the corresponding index as unique.

But do that only if you are sure that the key is indeed unique, as it imposes a uniqueness constraint on the indexed collection. An attempt to modify the indexed collection violating the uniqueness throws an System.InvalidOperationException.

For a subindex, uniqueness means that any given pair of parent key and subindex key values uniquely determines an item in the indexed collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SubindexCollection<T> Class
SubindexCollection<T> Members
Overload List

Clear Method
Clears the collection of all subindexes. All subindexes are detached from the parent and destroyed.

Syntax

Visual Basic (Declaration)
`Public Sub Clear()`
C#
`public void Clear()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SubindexCollection<T> Class
SubindexCollection<T> Members

Contains Method
Determines whether a subindex with the specified key selector exists in the collection.

Overload List

Overload	Description
Contains(LambdaExpression)	Determines whether a subindex with the specified key selector exists in the collection.

Contains<TKey>(Expression<Func<T,TKey>>)	Determines whether a subindex with the specified key selector exists in the collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SubindexCollection<T> Class
SubindexCollection<T> Members

Contains(LambdaExpression) Method
Key selector expression of a subindex, see KeySelector.

Determines whether a subindex with the specified key selector exists in the collection.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function Contains( _
   ByVal keySelector As LambdaExpression _
) As Boolean
```

C#

```
public bool Contains(
   LambdaExpression keySelector
)
```

Parameters

keySelector

Key selector expression of a subindex, see KeySelector.

Return Value

true if a subindex with the specified key selector is found in the collection; otherwise, **false**.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SubindexCollection<T> Class
SubindexCollection<T> Members
Overload List

Contains<TKey>(Expression<Func<T,TKey>>) Method
The type of the subindex key.

Key selector expression of a subindex, see KeySelector.

Determines whether a subindex with the specified key selector exists in the collection.

Syntax

Visual Basic (Declaration)

`Public Overloads Function Contains(Of TKey)( _`
`  ByVal keySelector As Expression(Of Func(Of T,TKey)) _`
`) As Boolean`

C#

`public bool Contains<TKey>(`
`  Expression<Func<T,TKey>> keySelector`
`)`

Parameters

keySelector

Key selector expression of a subindex, see KeySelector.

Type Parameters

TKey

The type of the subindex key.

Return Value

true if a subindex with the specified key selector is found in the collection; otherwise, **false**.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SubindexCollection<T> Class
SubindexCollection<T> Members
Overload List

Find Method
Finds a subindex in the collection by its key selector.

Overload List

Overload	Description
Find<TKey>(Expression<Func<T,TKey>>)	Finds a subindex in the collection by its key selector.
Find(LambdaExpression)	Finds a subindex in the collection by its key selector.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SubindexCollection<T> Class
SubindexCollection<T> Members

The type of the subindex key.

Key selector expression of an subindex, see KeySelector.

Finds a subindex in the collection by its key selector.

Syntax

Visual Basic (Declaration)

`Public Overloads Function Find(Of TKey)( _`
`  ByVal keySelector As Expression(Of Func(Of T,TKey)) _`
`) As Subindex(Of T,TKey)`

C#

`public Subindex<T,TKey> Find<TKey>(`
`  Expression<Func<T,TKey>> keySelector`
`)`

Parameters

keySelector

Key selector expression of an subindex, see KeySelector.

Type Parameters

TKey

The type of the subindex key.

Return Value

A subindex with the given key selector, if it is found; otherwise, **null**.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SubindexCollection<T> Class
SubindexCollection<T> Members
Overload List

Find(LambdaExpression) Method

Key selector expression of an subindex, see KeySelector.

Finds a subindex in the collection by its key selector.

Syntax

Visual Basic (Declaration)

`Public Overloads Function Find( _`
`  ByVal keySelector As LambdaExpression _`
`) As Subindex(Of T)`

C#

`public Subindex<T> Find(`
`  LambdaExpression keySelector`
`)`

Parameters

keySelector

Key selector expression of an subindex, see KeySelector.

Return Value

A subindex with the given key selector, if it is found; otherwise, **null**.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SubindexCollection<T> Class
SubindexCollection<T> Members
Overload List

GetEnumerator Method
Returns an enumerator that iterates through the SubindexCollection<T>.

Syntax

Visual Basic (Declaration)
`Public Function GetEnumerator() As IEnumerator(Of Subindex(Of T))`
C#
`public IEnumerator<Subindex<T>> GetEnumerator()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SubindexCollection<T> Class
SubindexCollection<T> Members

Remove Method
Removes a subindex from the collection.

Overload List

Overload	Description
Remove(Subindex<T>)	Removes a subindex from the collection.
Remove(LambdaExpression)	Removes a subindex from the collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SubindexCollection<T> Class
SubindexCollection<T> Members

Remove(Subindex<T>) Method
The subindex to remove.

Removes a subindex from the collection.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function Remove( _
   ByVal definition As Subindex(Of T) _
) As Boolean
```

C#

```
public bool Remove(
   Subindex<T> definition
)
```

Parameters

definition

The subindex to remove.

Return Value

true if a subindex has been removed; **false** if the subindex does not belong to this collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SubindexCollection<T> Class
SubindexCollection<T> Members
Overload List

Remove(LambdaExpression) Method
Key selector expression of a subindex, see KeySelector.

Removes a subindex from the collection.

Syntax

Visual Basic (Declaration)

`Public Overloads Function Remove( _`
`  ByVal keySelector As LambdaExpression _`
`) As Boolean`

C#

`public bool Remove(`
`  LambdaExpression keySelector`
`)`

Parameters

keySelector

Key selector expression of a subindex, see KeySelector.

Return Value

true if a subindex has been removed; **false** if there is no subindex with the given key selector in the collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SubindexCollection<T> Class
SubindexCollection<T> Members
Overload List

Properties
For a list of all members of this type, see SubindexCollection<T> members.

Public Properties

	Name	Description
	Count	Gets the number of subindexes in the collection.
	Item	Gets the subindex object at the specified ordinal position in the collection.

Top

See Also

Reference

SubindexCollection<T> Class
C1.LiveLinq.Indexing Namespace

Count Property
Gets the number of subindexes in the collection.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Property Count As Integer
```

C#

```
public int Count {get;}
```

Property Value

The number of subindexes in the collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SubindexCollection<T> Class
SubindexCollection<T> Members

Item Property
The zero-based ordinal position of the subindex to find

Gets the subindex object at the specified ordinal position in the collection.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Default Property Item( _
   ByVal ordinal As Integer _
) As Subindex(Of T)
```

C#

```
public Subindex<T> this[
   int ordinal
]; {get;}
```

Parameters

ordinal

The zero-based ordinal position of the subindex to find

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SubindexCollection<T> Class
SubindexCollection<T> Members

Interfaces

IIndexedSource<T>
The type of the elements of the collection.

Represents an indexed collection.

Object Model

Syntax

Visual Basic (Declaration)
`Public Interface IIndexedSource(Of T)`
C#
`public interface IIndexedSource<T>`

Type Parameters

T

The type of the elements of the collection.

Remarks

An indexed collection has a collection of indexes, ScannerCollection<T> that are maintained up-to-date on every change made to the collection.

This interface is implemented by all main LiveLinq collection classes: C1.LiveLinq.Collections.IndexedCollection<T>, C1.LiveLinq.AdoNet.IndexedDataTable<TRow>, C1.LiveLinq.LiveViews.View<T>.

You need to implement this interface only if you want to define your own indexable collection classes and then only if they don't inherit from C1.LiveLinq.Collections.IndexedCollection<T>, see LiveLinq to Objects: IndexedCollection(T) and other collection classes.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IIndexedSource<T> Members
C1.LiveLinq.Indexing Namespace

Overview
The type of the elements of the collection.

Represents an indexed collection.

Object Model

Syntax

Visual Basic (Declaration)
`Public Interface IIndexedSource(Of T)`
C#
`public interface IIndexedSource<T>`

Type Parameters

T

The type of the elements of the collection.

Remarks

An indexed collection has a collection of indexes, ScannerCollection<T> that are maintained up-to-date on every change made to the collection.

This interface is implemented by all main LiveLinq collection classes: C1.LiveLinq.Collections.IndexedCollection<T>, C1.LiveLinq.AdoNet.IndexedDataTable<TRow>, C1.LiveLinq.LiveViews.View<T>.

You need to implement this interface only if you want to define your own indexable collection classes and then only if they don't inherit from C1.LiveLinq.Collections.IndexedCollection<T>, see LiveLinq to Objects: IndexedCollection(T) and other collection classes.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IIndexedSource<T> Members
C1.LiveLinq.Indexing Namespace

Members
Properties

The following tables list the members exposed by IIndexedSource<T>.

Public Properties

	Name	Description
	Indexes	Gets the collection of indexes attached to this collection.

Top

See Also

Reference

IIndexedSource<T> Interface
C1.LiveLinq.Indexing Namespace

Properties
For a list of all members of this type, see IIndexedSource<T> members.

Public Properties

	Name	Description

	Indexes	Gets the collection of indexes attached to this collection.

Top

See Also

Reference

IIndexedSource<T> Interface
C1.LiveLinq.Indexing Namespace

Indexes Property
Gets the collection of indexes attached to this collection.

Syntax

Visual Basic (Declaration)
`ReadOnly Property Indexes As ScannerCollection(Of T)`
C#
`ScannerCollection<T> Indexes {get;}`

Property Value

A collection of indexes attached to this collection. If this is an independent collection, not the result of a LiveLinq indexing search, then its Indexes collection contains Index<T> objects. Otherwise, that is, if it is the result of an indexing search operation such as Index.Find and others, it contains subindexes implementing C1.LiveLinq.Indexing.Search.IIndexScanner<T>.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IIndexedSource<T> Interface
IIndexedSource<T> Members

C1.LiveLinq.Indexing.Search Namespace

Overview

Classes

	Class	Description
	GroupingQuery<T>	Represents a collection of IndexedGroup<T>, groups of elements with the same key, resulting from a search operation with grouping. This class has a derived class GroupingQuery<TKey,T> with specific type of the key used for the index search.
	GroupingQuery<TKey,T>	Represents a collection of IndexedGroup<TKey,T>, groups of elements with the same key, resulting from a search operation with grouping.
	IndexedGroup<T>	Represents a group of elements with the same key belonging to a collection of groups resulting from a search operation with grouping. This class has a derived class IndexedGroup<TKey,T> with specific key type.
	IndexedGroup<TKey,T>	Represents a group of elements with the same key belonging to a collection of groups resulting from a search operation with grouping.
	IndexQuery<T>	Represents a collection that is the result of an index search. Objects of this class are returned by the IIndexScanner<T> search methods. This class has a derived class IndexQuery<T,TKey> with specific type of the key used for the index search.
	IndexQuery<T,TKey>	Represents a collection that is the result of an index search. Objects of this class are returned by the IIndexScanner<T,TKey> search methods.

Interfaces

	Interface	Description

	IIndexScanner<T>	Represents an index or a subindex in its capacity of scanning through data. Provides methods for searching data items.
	IIndexScanner<T,TKey>	Represents an index or a subindex in its capacity of scanning through data. Provides methods for searching data items.

Enumerations

	Enumeration	Description
	JoinOperator	A comparison operator to match elements in a join operation.

See Also

Reference

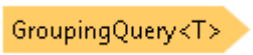
C1.LiveLinq.4 Assembly

Classes

GroupingQuery<T>
The type of the elements of the indexed collection.

Represents a collection of IndexedGroup<T>, groups of elements with the same key, resulting from a search operation with grouping. This class has a derived class GroupingQuery<TKey,T> with specific type of the key used for the index search.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit Class GroupingQuery(Of T)`
C#
`public abstract class GroupingQuery<T>`

Type Parameters

T

The type of the elements of the indexed collection.

Remarks

The result of any index search operation, IndexQuery<T> can be grouped by applying IndexQuery<T>.GroupByUntypedKey. It is grouped by the key that was used to perform the search operation. For example, `customers.Indexes(c => c.City).All().GroupByUntypedKey(); customers.Indexes(c => c.City).FindBetween("A", "Z").GroupByUntypedKey();`

Inheritance Hierarchy

System.Object
C1.LiveLinq.Indexing.Search.GroupingQuery<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

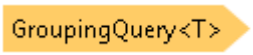
Reference

GroupingQuery<T> Members
C1.LiveLinq.Indexing.Search Namespace
IndexedGroup<TKey,T> Class

Overview
The type of the elements of the indexed collection.

Represents a collection of IndexedGroup<T>, groups of elements with the same key, resulting from a search operation with grouping. This class has a derived class GroupingQuery<TKey,T> with specific type of the key used for the index search.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit Class GroupingQuery(Of T)`
C#
`public abstract class GroupingQuery<T>`

Type Parameters

T

The type of the elements of the indexed collection.

Remarks

The result of any index search operation, IndexQuery<T> can be grouped by applying IndexQuery<T>.GroupByUntypedKey. It is grouped by the key that was used to perform the search operation. For example, `customers.Indexes(c => c.City).All().GroupByUntypedKey(); customers.Indexes(c => c.City).FindBetween("A", "Z").GroupByUntypedKey();`

Inheritance Hierarchy

System.Object
  C1.LiveLinq.Indexing.Search.GroupingQuery<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

GroupingQuery<T> Members
C1.LiveLinq.Indexing.Search Namespace
IndexedGroup<TKey,T> Class

Members
Methods

The following tables list the members exposed by GroupingQuery<T>.

Public Methods

	Name	Description
	GetEnumerator	Returns an enumerator that iterates through the GroupingQuery<T>.

Top

See Also

Reference

GroupingQuery<T> Class
C1.LiveLinq.Indexing.Search Namespace
IndexedGroup<TKey,T> Class

Methods
For a list of all members of this type, see GroupingQuery<T> members.

Public Methods

	Name	Description
	GetEnumerator	Returns an enumerator that iterates through the GroupingQuery<T>.

Top

See Also

Reference

GroupingQuery<T> Class
C1.LiveLinq.Indexing.Search Namespace
IndexedGroup<TKey,T> Class

GetEnumerator Method
Returns an enumerator that iterates through the GroupingQuery<T>.

Syntax

Visual Basic (Declaration)	

```
Public MustOverride Function GetEnumerator() As IEnumerator(Of IndexedGroup(Of T))
```

C#

```
public abstract IEnumerator<IndexedGroup<T>> GetEnumerator()
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

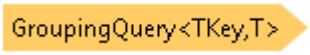
GroupingQuery<T> Class
GroupingQuery<T> Members

GroupingQuery<TKey,T>
The type of the index key.

The type of the elements of the indexed collection.

Represents a collection of IndexedGroup<TKey,T>, groups of elements with the same key, resulting from a search operation with grouping.

Object Model



Syntax

Visual Basic (Declaration)

```
Public MustInherit Class GroupingQuery
    (Of TKey,T)
```

C#

```
public abstract class GroupingQuery<TKey,T>
```

Type Parameters

TKey

The type of the index key.

T

The type of the elements of the indexed collection.

Remarks

The result of any index search operation, IndexQuery<T,TKey> can be grouped by applying IndexQuery<T,TKey>.GroupByKey. It is grouped by the key that was used to perform the search operation. For example, `customers.Indexes(c => c.City).All().GroupByKey(); customers.Indexes(c => c.City).FindBetween("A", "Z").GroupByKey();`

Inheritance Hierarchy

System.Object
  C1.LiveLinq.Indexing.Search.GroupingQuery<TKey,T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

GroupingQuery<TKey,T> Members
C1.LiveLinq.Indexing.Search Namespace
IndexedGroup<T> Class

Overview
The type of the index key.

The type of the elements of the indexed collection.

Represents a collection of IndexedGroup<TKey,T>, groups of elements with the same key, resulting from a search operation with grouping.

Object Model

GroupingQuery<TKey,T>

Syntax

Visual Basic (Declaration)
`Public MustInherit Class GroupingQuery` `    (Of TKey,T)`
C#
`public abstract class GroupingQuery<TKey,T>`

Type Parameters

TKey

The type of the index key.

T

The type of the elements of the indexed collection.

Remarks

The result of any index search operation, IndexQuery<T,TKey> can be grouped by applying IndexQuery<T,TKey>.GroupByKey. It is grouped by the key that was used to perform the search operation. For example, `customers.Indexes(c => c.City).All().GroupByKey(); customers.Indexes(c => c.City).FindBetween("A", "Z").GroupByKey();`

Inheritance Hierarchy

System.Object
  C1.LiveLinq.Indexing.Search.GroupingQuery<TKey,T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

GroupingQuery<TKey,T> Members
C1.LiveLinq.Indexing.Search Namespace
IndexedGroup<T> Class

Members
Methods

The following tables list the members exposed by GroupingQuery<TKey,T>.

Public Methods

	Name	Description
	GetEnumerator	Returns an enumerator that iterates through the GroupingQuery<TKey,T>.

Top

See Also

Reference

GroupingQuery<TKey,T> Class
C1.LiveLinq.Indexing.Search Namespace
IndexedGroup<T> Class

Methods
For a list of all members of this type, see GroupingQuery<TKey,T> members.

Public Methods

	Name	Description
	GetEnumerator	Returns an enumerator that iterates through the GroupingQuery<TKey,T>.

Top

See Also

Reference

GroupingQuery<TKey,T> Class
C1.LiveLinq.Indexing.Search Namespace
IndexedGroup<T> Class

GetEnumerator Method
Returns an enumerator that iterates through the GroupingQuery<TKey,T>.

Syntax

Visual Basic (Declaration)
`Public MustOverride Function GetEnumerator() As IEnumerator(Of IndexedGroup(Of TKey,T))`
C#
`public abstract IEnumerator<IndexedGroup<TKey,T>> GetEnumerator()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

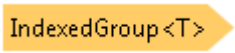
Reference

GroupingQuery<TKey,T> Class
GroupingQuery<TKey,T> Members

IndexedGroup<T>
The type of the elements of the indexed collection.

Represents a group of elements with the same key belonging to a collection of groups resulting from a search operation with grouping. This class has a derived class IndexedGroup<TKey,T> with specific key type.

Object Model



Syntax

Visual Basic (Declaration)	

`Public MustInherit Class IndexedGroup(Of T)`
`  Inherits IndexQuery(Of T)`
`  Implements C1.LiveLinq.Indexing.IIndexedSource(Of T)`	

C#	
`public abstract class IndexedGroup<T> : IndexQuery<T>, C1.LiveLinq.Indexing.IIndexedSource<T>`	

Type Parameters

T

The type of the elements of the indexed collection.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.Indexing.Search.IndexQuery<T>
    C1.LiveLinq.Indexing.Search.IndexedGroup<T>
      C1.LiveLinq.Indexing.Search.IndexedGroup<TKey,T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedGroup<T> Members
C1.LiveLinq.Indexing.Search Namespace
GroupingQuery<T> Class
IndexedGroup<TKey,T> Class

Overview
The type of the elements of the indexed collection.

Represents a group of elements with the same key belonging to a collection of groups resulting from a search operation with grouping. This class has a derived class IndexedGroup<TKey,T> with specific key type.

Object Model

Syntax

Visual Basic (Declaration)

`Public MustInherit Class IndexedGroup(Of T)`
`  Inherits IndexQuery(Of T)`
`  Implements C1.LiveLinq.Indexing.IIndexedSource(Of T)`

C#
`public abstract class IndexedGroup<T> : IndexQuery<T>, C1.LiveLinq.Indexing.IIndexedSource<T>`

Type Parameters

T

The type of the elements of the indexed collection.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.Indexing.Search.IndexQuery<T>
    C1.LiveLinq.Indexing.Search.IndexedGroup<T>
      C1.LiveLinq.Indexing.Search.IndexedGroup<TKey,T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedGroup<T> Members
C1.LiveLinq.Indexing.Search Namespace
GroupingQuery<T> Class
IndexedGroup<TKey,T> Class

Members

Properties Methods

The following tables list the members exposed by IndexedGroup<T>.

Public Properties

	Name	Description
	Indexes	The collection of subindexes for this IndexQuery<T>. (Inherited from C1.LiveLinq.Indexing.Search.IndexQuery<T>)
	Key	Gets the key of the IndexedGroup<T>.
	KeyType	Gets the type of the key of the IndexedGroup<T>.

Top

Public Methods

	Name	Description
	GetEnumerator	Returns an enumerator that iterates through the IndexQuery<T>. (Inherited from C1.LiveLinq.Indexing.Search.IndexQuery<T>)
	GroupByUntypedKey	Groups the collection of search results by its search key. (Inherited from C1.LiveLinq.Indexing.Search.IndexQuery<T>)
	Subindex	Overloaded. Used to apply subindex search to the result of a search operation. (Inherited from C1.LiveLinq.Indexing.Search.IndexQuery<T>)

Top

See Also

Reference

IndexedGroup<T> Class
C1.LiveLinq.Indexing.Search Namespace
GroupingQuery<T> Class
IndexedGroup<TKey,T> Class

Properties
For a list of all members of this type, see IndexedGroup<T> members.

Public Properties

	Name	Description
	Indexes	The collection of subindexes for this IndexQuery<T>. (Inherited from C1.LiveLinq.Indexing.Search.IndexQuery<T>)
	Key	Gets the key of the IndexedGroup<T>.
	KeyType	Gets the type of the key of the IndexedGroup<T>.

Top

See Also

Reference

IndexedGroup<T> Class
C1.LiveLinq.Indexing.Search Namespace
GroupingQuery<T> Class
IndexedGroup<TKey,T> Class

Key Property
Gets the key of the IndexedGroup<T>.

Syntax

Visual Basic (Declaration)	
`Public ReadOnly Property Key As Object`	
C#	

```
public object Key {get;}
```

Property Value

The key of the IndexedGroup<T>.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedGroup<T> Class
IndexedGroup<T> Members

KeyType Property
Gets the type of the key of the IndexedGroup<T>.

Syntax

Visual Basic (Declaration)
`Public MustOverride ReadOnly Property KeyType As Type`
C#
`public abstract Type KeyType {get;}`

Property Value

Type of the key of the IndexedGroup<T>.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

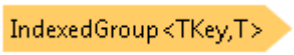
IndexedGroup<T> Class
IndexedGroup<T> Members

IndexedGroup<TKey,T>
The type of the index key.

The type of the elements of the indexed collection.

Represents a group of elements with the same key belonging to a collection of groups resulting from a search operation with grouping.

Object Model



Syntax

Visual Basic (Declaration)

`Public MustInherit Class IndexedGroup`
`  (Of TKey,T)`
`  Inherits IndexedGroup(Of T)`
`  Implements C1.LiveLinq.Indexing.IIndexedSource(Of T)`

C#
`public abstract class IndexedGroup<TKey,T> : IndexedGroup<T>,` `C1.LiveLinq.Indexing.IIndexedSource<T>`

Type Parameters

TKey

The type of the index key.

T

The type of the elements of the indexed collection.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.Indexing.Search.IndexQuery<T>

C1.LiveLinq.Indexing.Search.IndexedGroup<T>

C1.LiveLinq.Indexing.Search.IndexedGroup<TKey,T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedGroup<TKey,T> Members
C1.LiveLinq.Indexing.Search Namespace
GroupingQuery<TKey,T> Class
IndexedGroup<T> Class

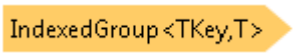
Overview

The type of the index key.

The type of the elements of the indexed collection.

Represents a group of elements with the same key belonging to a collection of groups resulting from a search operation with grouping.

Object Model



Syntax

Visual Basic (Declaration)	

`Public MustInherit Class IndexedGroup`
`  (Of TKey,T)`
`  Inherits IndexedGroup(Of T)`
`  Implements C1.LiveLinq.Indexing.IIndexedSource(Of T)`	

C#	
`public abstract class IndexedGroup<TKey,T> : IndexedGroup<T>,` `C1.LiveLinq.Indexing.IIndexedSource<T>`	

Type Parameters

TKey

The type of the index key.

T

The type of the elements of the indexed collection.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.Indexing.Search.IndexQuery<T>
    C1.LiveLinq.Indexing.Search.IndexedGroup<T>
      C1.LiveLinq.Indexing.Search.IndexedGroup<TKey,T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedGroup<TKey,T> Members
C1.LiveLinq.Indexing.Search Namespace
GroupingQuery<TKey,T> Class
IndexedGroup<T> Class

Members
Properties Methods

The following tables list the members exposed by IndexedGroup<TKey,T>.

Public Properties

	Name	Description
	Indexes	The collection of subindexes for this IndexQuery<T>. (Inherited from C1.LiveLinq.Indexing.Search.IndexQuery<T>)

	Key	Gets the key of the IndexedGroup<TKey,T>.
	KeyType	Overridden. Gets the type of the key of the IndexedGroup<TKey,T>.

Top

Public Methods

	Name	Description
	GetEnumerator	Returns an enumerator that iterates through the IndexQuery<T>. (Inherited from C1.LiveLinq.Indexing.Search.IndexQuery<T>)
	GroupByUntypedKey	Groups the collection of search results by its search key. (Inherited from C1.LiveLinq.Indexing.Search.IndexQuery<T>)
	Subindex	Overloaded. (Inherited from C1.LiveLinq.Indexing.Search.IndexQuery<T>)

Top

See Also

Reference

IndexedGroup<TKey,T> Class
C1.LiveLinq.Indexing.Search Namespace
GroupingQuery<TKey,T> Class
IndexedGroup<T> Class

Properties
For a list of all members of this type, see IndexedGroup<TKey,T> members.

Public Properties

	Name	Description
	Indexes	The collection of subindexes for this IndexQuery<T>. (Inherited from C1.LiveLinq.Indexing.Search.IndexQuery<T>)

	Key	Gets the key of the IndexedGroup<TKey,T>.
	KeyType	Overridden. Gets the type of the key of the IndexedGroup<TKey,T>.

Top

See Also

Reference

IndexedGroup<TKey,T> Class
C1.LiveLinq.Indexing.Search Namespace
GroupingQuery<TKey,T> Class
IndexedGroup<T> Class

Key Property
Gets the key of the IndexedGroup<TKey,T>.

Syntax

Visual Basic (Declaration)
`Public MustOverride Shadows ReadOnly Property Key As TKey`
C#
`public abstract new TKey Key {get;}`

Property Value

The key of the IndexedGroup<TKey,T>.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexedGroup<TKey,T> Class
IndexedGroup<TKey,T> Members

KeyType Property
Gets the type of the key of the IndexedGroup<TKey,T>.

Syntax

Visual Basic (Declaration)
`Public Overrides ReadOnly Property KeyType As Type`
C#
`public override Type KeyType {get;}`

Property Value

The same type as the *TKey* type parameter of the class.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

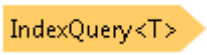
Reference

IndexedGroup<TKey,T> Class
IndexedGroup<TKey,T> Members

IndexQuery<T>
The type of the elements of the indexed collection.

Represents a collection that is the result of an index search. Objects of this class are returned by the IIndexScanner<T> search methods. This class has a derived class IndexQuery<T,TKey> with specific type of the key used for the index search.

Object Model



Syntax

Visual Basic (Declaration)	
`Public MustInherit Class IndexQuery(Of T)` `  Implements C1.LiveLinq.Indexing.IIndexedSource(Of T)`	
C#	
`public abstract class IndexQuery<T> : C1.LiveLinq.Indexing.IIndexedSource<T>`	

Type Parameters

T

The type of the elements of the indexed collection.

Inheritance Hierarchy

System.Object
C1.LiveLinq.Indexing.Search.IndexQuery<T>
C1.LiveLinq.Indexing.Search.IndexedGroup<T>
C1.LiveLinq.Indexing.Search.IndexQuery<T,TKey>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexQuery<T> Members
C1.LiveLinq.Indexing.Search Namespace
IndexQuery<T,TKey> Class

Overview
The type of the elements of the indexed collection.

Represents a collection that is the result of an index search. Objects of this class are returned by the IIndexScanner<T> search methods. This class has a derived class IndexQuery<T,TKey> with specific type of the key used for the index search.

Object Model

IndexQuery<T>

Syntax

Visual Basic (Declaration)
`Public MustInherit Class IndexQuery(Of T)` `  Implements C1.LiveLinq.Indexing.IIndexedSource(Of T)`
C#
`public abstract class IndexQuery<T> : C1.LiveLinq.Indexing.IIndexedSource<T>`

Type Parameters

T

The type of the elements of the indexed collection.

Inheritance Hierarchy

System.Object
C1.LiveLinq.Indexing.Search.IndexQuery<T>
C1.LiveLinq.Indexing.Search.IndexedGroup<T>
C1.LiveLinq.Indexing.Search.IndexQuery<T,TKey>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexQuery<T> Members
C1.LiveLinq.Indexing.Search Namespace
IndexQuery<T,TKey> Class

Members
Properties Methods

The following tables list the members exposed by IndexQuery<T>.

Public Properties

	Name	Description
	Indexes	The collection of subindexes for this IndexQuery<T>.

Top

Public Methods

	Name	Description
	GetEnumerator	Returns an enumerator that iterates through the IndexQuery<T>.
	GroupByUntypedKey	Groups the collection of search results by its search key.
	Subindex	Overloaded. Used to apply subindex search to the result of a search operation.

Top

See Also

Reference

IndexQuery<T> Class
C1.LiveLinq.Indexing.Search Namespace
IndexQuery<T,TKey> Class

Methods
For a list of all members of this type, see IndexQuery<T> members.

Public Methods

	Name	Description
	GetEnumerator	Returns an enumerator that iterates through the IndexQuery<T>.
	GroupByUntypedKey	Groups the collection of search results by its search key.

	Subindex	Overloaded. Used to apply subindex search to the result of a search operation.

Top

See Also

Reference

IndexQuery<T> Class
C1.LiveLinq.Indexing.Search Namespace
IndexQuery<T,TKey> Class

GetEnumerator Method
Returns an enumerator that iterates through the IndexQuery<T>.

Syntax

Visual Basic (Declaration)

```
Public MustOverride Function GetEnumerator() As IEnumerator(Of T)
```

C#

```
public abstract IEnumerator<T> GetEnumerator()
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexQuery<T> Class
IndexQuery<T> Members

GroupByUntypedKey Method
Groups the collection of search results by its search key.

Syntax

Visual Basic (Declaration)
`Public Overridable Function GroupByUntypedKey() As GroupingQuery(Of T)`
C#
`public virtual GroupingQuery<T> GroupByUntypedKey()`

Return Value

Search result collection grouped by its search key.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexQuery<T> Class
IndexQuery<T> Members
IndexedGroup<T> Class
GroupByKey Method

Subindex Method
Used to apply subindex search to the result of a search operation.

Overload List

Overload	Description
Subindex<TKey>(Subindex<T,TKey>)	Used to apply subindex search to the result of a search operation.
Subindex(Subindex<T>)	Used to apply subindex search to the result of a search operation.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexQuery<T> Class
IndexQuery<T> Members

Subindex<TKey>(Subindex<T,TKey>) Method
Type of the subindex key.

Subindex to use for narrowing the search.

Used to apply subindex search to the result of a search operation.

Syntax

Visual Basic (Declaration)

```
Public Overloads Overridable Function Subindex(Of TKey)( _
   ByVal definition As Subindex(Of T,TKey) _
) As IIndexScanner(Of T,TKey)
```

C#

```
public virtual IIndexScanner<T,TKey> Subindex<TKey>(
   Subindex<T,TKey> definition
)
```

Parameters

definition

Subindex to use for narrowing the search.

Type Parameters

TKey

Type of the subindex key.

Return Value

An IIndexScanner<T,TKey> collection indexed by the subindex that can be used to perform search operations narrowing the collection.

Remarks

A subindex can be used to further narrow the result of a search operation, if the corresponding subindex exists in the index or subindex used to perform that search operation. For example, `var idxByCity = customers.Indexes(c => c.City); var subindexByContactTitle = idxByCity.Subindexes(c => c.ContactTitle); var ownersInLondon = idxByCity.Find("London").Subindex(subindexByContactTitle).Find("Owner");`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexQuery<T> Class
IndexQuery<T> Members
Overload List

Subindex(Subindex<T>) Method
Subindex to use for narrowing the search.

Used to apply subindex search to the result of a search operation.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function Subindex( _
   ByVal definition As Subindex(Of T) _
) As IIndexScanner(Of T)
```

C#

```
public IIndexScanner<T> Subindex(
   Subindex<T> definition
```

)

Parameters

definition

Subindex to use for narrowing the search.

Return Value

An IIndexScanner<T> collection indexed by the subindex that can be used to perform search operations narrowing the collection.

Remarks

A subindex can be used to further narrow the result of a search operation, if the corresponding subindex exists in the index or subindex used to perform that search operation. For example, `var idxByCity = customers.Indexes(c => c.City); var subindexByContactTitle = idxByCity.Subindexes(c => c.ContactTitle); var ownersInLondon = idxByCity.Find("London").Subindex<string>(subindexByContactTitle).Find("Owner");`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexQuery<T> Class
IndexQuery<T> Members
Overload List

Properties
For a list of all members of this type, see IndexQuery<T> members.

Public Properties

	Name	Description

	Indexes	The collection of subindexes for this IndexQuery<T>.

Top

See Also

Reference

IndexQuery<T> Class
C1.LiveLinq.Indexing.Search Namespace
IndexQuery<T,TKey> Class

Indexes Property
The collection of subindexes for this IndexQuery<T>.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Property Indexes As ScannerCollection(Of T)
```

C#

```
public ScannerCollection<T> Indexes {get;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexQuery<T> Class
IndexQuery<T> Members

IndexQuery<T,TKey>
The type of the elements of the indexed collection.

The type of the index key.

Represents a collection that is the result of an index search. Objects of this class are returned by the IIndexScanner<T,TKey> search methods.

Object Model

IndexQuery<T,TKey>

Syntax

Visual Basic (Declaration)	

`Public MustInherit Class IndexQuery`
`  (Of T,TKey)`
`  Inherits IndexQuery(Of T)`
`  Implements C1.LiveLinq.Indexing.IIndexedSource(Of T)`	

C#	
`public abstract class IndexQuery<T,TKey> : IndexQuery<T>,` `C1.LiveLinq.Indexing.IIndexedSource<T>`	

Type Parameters

T

The type of the elements of the indexed collection.

TKey

The type of the index key.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.Indexing.Search.IndexQuery<T>
    C1.LiveLinq.Indexing.Search.IndexQuery<T,TKey>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

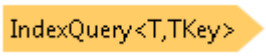
IndexQuery<T,TKey> Members
C1.LiveLinq.Indexing.Search Namespace
IndexQuery<T> Class

Overview
The type of the elements of the indexed collection.

The type of the index key.

Represents a collection that is the result of an index search. Objects of this class are returned by the IIndexScanner<T,TKey> search methods.

Object Model



Syntax

Visual Basic (Declaration)	

`Public MustInherit Class IndexQuery`
`  (Of T,TKey)`
`  Inherits IndexQuery(Of T)`
`  Implements C1.LiveLinq.Indexing.IIndexedSource(Of T)`	

C#	
`public abstract class IndexQuery<T,TKey> : IndexQuery<T>,` `C1.LiveLinq.Indexing.IIndexedSource<T>`	

Type Parameters

T

The type of the elements of the indexed collection.

TKey

The type of the index key.

Inheritance Hierarchy

System.Object
C1.LiveLinq.Indexing.Search.IndexQuery<T>
C1.LiveLinq.Indexing.Search.IndexQuery<T,TKey>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexQuery<T,TKey> Members
C1.LiveLinq.Indexing.Search Namespace
IndexQuery<T> Class

Members
Properties Methods

The following tables list the members exposed by IndexQuery<T,TKey>.

Public Properties

	Name	Description
	Indexes	The collection of subindexes for this IndexQuery<T>. (Inherited from C1.LiveLinq.Indexing.Search.IndexQuery<T>)

Top

Public Methods

	Name	Description
	GetEnumerator	Returns an enumerator that iterates through the IndexQuery<T>. (Inherited from C1.LiveLinq.Indexing.Search.IndexQuery<T>)
	GroupByKey	Groups the collection of search results by its search key.

	GroupByUntypedKey	Overridden. Groups the collection of search results by its search key.
	Subindex	Overloaded. Used to apply subindex search to the result of a search operation. (Inherited from C1.LiveLinq.Indexing.Search.IndexQuery<T>)

Top

See Also

Reference

IndexQuery<T,TKey> Class
C1.LiveLinq.Indexing.Search Namespace
IndexQuery<T> Class

Methods
For a list of all members of this type, see IndexQuery<T,TKey> members.

Public Methods

	Name	Description
	GetEnumerator	Returns an enumerator that iterates through the IndexQuery<T>. (Inherited from C1.LiveLinq.Indexing.Search.IndexQuery<T>)
	GroupByKey	Groups the collection of search results by its search key.
	GroupByUntypedKey	Overridden. Groups the collection of search results by its search key.
	Subindex	Overloaded. Used to apply subindex search to the result of a search operation. (Inherited from C1.LiveLinq.Indexing.Search.IndexQuery<T>)

Top

See Also

Reference

IndexQuery<T,TKey> Class
C1.LiveLinq.Indexing.Search Namespace
IndexQuery<T> Class

GroupByKey Method

Groups the collection of search results by its search key.

Syntax

Visual Basic (Declaration)
`Public Overridable Function GroupByKey() As GroupingQuery(Of TKey,T)`
C#
`public virtual GroupingQuery<TKey,T> GroupByKey()`

Return Value

Search result collection grouped by its search key.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexQuery<T,TKey> Class
IndexQuery<T,TKey> Members
IndexedGroup<TKey,T> Class
GroupByUntypedKey Method

GroupByUntypedKey Method

Groups the collection of search results by its search key.

Syntax

Visual Basic (Declaration)
`Public Overrides Function GroupByUntypedKey() As GroupingQuery(Of T)`
C#
`public override GroupingQuery<T> GroupByUntypedKey()`

Return Value

Search result collection grouped by its search key.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IndexQuery<T,TKey> Class
IndexQuery<T,TKey> Members
IndexedGroup<T> Class
GroupByKey Method

Enumerations

JoinOperator
A comparison operator to match elements in a join operation.

Syntax

Visual Basic (Declaration)
`Public Enum JoinOperator` `  Inherits System.Enum`
C#
`public enum JoinOperator : System.Enum`

Members

Member	Description
Equal	**a** is equal to **b** (**a** == **b**)
Greater	**a** is greater than **b** (**a** > **b**)

GreaterOrEqual	**a** is greater than or equal to **b** (**a** >= **b**)
Less	**a** is less than **b** (**a** < **b**)
LessOrEqual	**a** is less than or equal to **b** (**a** <= **b**)

Inheritance Hierarchy

System.Object
  System.ValueType
    System.Enum
      C1.LiveLinq.Indexing.Search.JoinOperator

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

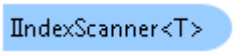
C1.LiveLinq.Indexing.Search Namespace

Interfaces

IIndexScanner<T>
The type of the elements of the indexed collection.

Represents an index or a subindex in its capacity of scanning through data. Provides methods for searching data items.

Object Model



Syntax

Visual Basic (Declaration)
`Public Interface IIndexScanner(Of T)`

C#

```
public interface IIndexScanner<T>
```

Type Parameters

T

The type of the elements of the indexed collection.

Remarks

This interface is implemented by C1.LiveLinq.Indexing.Index<T>. It is also used by subindexes, but there it is not directly implemented by C1.LiveLinq.Indexing.Subindex<T>, but rather returned by the IndexQuery<T>.Subindex method because it depends on the item found by an index or a subindex that is the parent of that subindex.

IIndexScanner<T> has a typed key counterpart IIndexScanner<T,TKey> that is used with typed key classes C1.LiveLinq.Indexing.Index<T,TKey> and C1.LiveLinq.Indexing.Subindex<T,TKey>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

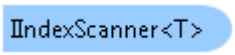
Reference

IIndexScanner<T> Members
C1.LiveLinq.Indexing.Search Namespace

Overview
The type of the elements of the indexed collection.

Represents an index or a subindex in its capacity of scanning through data. Provides methods for searching data items.

Object Model



Syntax

Visual Basic (Declaration)	
`Public Interface IIndexScanner(Of T)`	
C#	
`public interface IIndexScanner<T>`	

Type Parameters

T

The type of the elements of the indexed collection.

Remarks

This interface is implemented by C1.LiveLinq.Indexing.Index<T>. It is also used by subindexes, but there it is not directly implemented by C1.LiveLinq.Indexing.Subindex<T>, but rather returned by the IndexQuery<T>.Subindex method because it depends on the item found by an index or a subindex that is the parent of that subindex.

IIndexScanner<T> has a typed key counterpart IIndexScanner<T,TKey> that is used with typed key classes C1.LiveLinq.Indexing.Index<T,TKey> and C1.LiveLinq.Indexing.Subindex<T,TKey>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IIndexScanner<T> Members
C1.LiveLinq.Indexing.Search Namespace

Members
Properties Methods

The following tables list the members exposed by IIndexScanner<T>.

Public Properties

	Name	Description
	Definition	Gets an C1.LiveLinq.Indexing.Index<T> or a C1.LiveLinq.Indexing.Subindex<T> definition on which the scanner is based.
	KeyCount	Gets the number of distinct key values in all items of this collection.
	ParentScanner	Gets an index or a subindex scanner that is the parent of a subindex scanner.

Top

Public Methods

	Name	Description
	All	Gets all items in the indexed collection.
	ContainsKey	Returns a value that indicates whether the collection contains an item with the given key value.
	Find	Finds items with the specified key value.
	FindBetween	Finds items with key values in the interval between the specified values.
	FindGreater	Finds items with keys greater than the specified value.
	FindKeys	Finds items containing any of the specified key values.
	FindLess	Finds items with keys less than the specified value.
	FindStartingWith	Finds items with string key values starting with the specified string.
	GroupJoin	Overloaded. Correlates the items of this indexed collection with the items of another indexed collection and groups the results by the item of this collection.

	Join	Overloaded. Correlates the items of this indexed collection with the items of another sequence and returns the combined items with matching keys.

Top

See Also

Reference

IIndexScanner<T> Interface
C1.LiveLinq.Indexing.Search Namespace

Methods
For a list of all members of this type, see IIndexScanner<T> members.

Public Methods

	Name	Description
	All	Gets all items in the indexed collection.
	ContainsKey	Returns a value that indicates whether the collection contains an item with the given key value.
	Find	Finds items with the specified key value.
	FindBetween	Finds items with key values in the interval between the specified values.
	FindGreater	Finds items with keys greater than the specified value.
	FindKeys	Finds items containing any of the specified key values.
	FindLess	Finds items with keys less than the specified value.
	FindStartingWith	Finds items with string key values starting with the specified string.
	GroupJoin	Overloaded. Correlates the items of this indexed collection with the items of another indexed collection and groups the results by the item of this

		collection.
	Join	Overloaded. Correlates the items of this indexed collection with the items of another sequence and returns the combined items with matching keys.

Top

See Also

Reference

IIndexScanner<T> Interface
C1.LiveLinq.Indexing.Search Namespace

All Method
Specifies the order of the key values to sort the result.

Gets all items in the indexed collection.

Syntax

Visual Basic (Declaration)

```
Function All( _
   ByVal order As Order _
) As IndexQuery(Of T)
```

C#

```
IndexQuery<T> All(
   Order order
)
```

Parameters

order

Specifies the order of the key values to sort the result.

Return Value

An object enumerating all items of the collection in the specified order of their key values.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IIndexScanner<T> Interface
IIndexScanner<T> Members

ContainsKey Method
The key value to search for

Returns a value that indicates whether the collection contains an item with the given key value.

Syntax

Visual Basic (Declaration)

`Function ContainsKey( _`
`  ByVal key As Object _`
`) As Boolean`

C#

`bool ContainsKey(`
`  object key`
`)`

Parameters

key

The key value to search for

Return Value

true if the collection contains an element with the specified key value; otherwise, **false**.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IIndexScanner<T> Interface
IIndexScanner<T> Members

Find Method
The key value to search for.

Finds items with the specified key value.

Syntax

Visual Basic (Declaration)

`Function Find( _`
`  ByVal key As Object _`
`) As IndexQuery(Of T)`

C#

`IndexQuery<T> Find(`
`  object key`
`)`

Parameters

key

The key value to search for.

Return Value

An object enumerating items having the specified key value.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IIndexScanner<T> Interface
IIndexScanner<T> Members

FindBetween Method

Minimum key value to search for.

If **true**, the result includes items with the minimum key value.

Maximum key value to search for.

If **true**, the result includes items with the maximum key value.

Optionally specifies the order of the key values to sort the result (C1.LiveLinq.Order.**Unordered** if sorting is not required).

Finds items with key values in the interval between the specified values.

Syntax

Visual Basic (Declaration)

```
Function FindBetween( _
   ByVal min As Object, _
   ByVal minInclusive As Boolean, _
   ByVal max As Object, _
   ByVal maxInclusive As Boolean, _
   ByVal order As Order _
) As IndexQuery(Of T)
```

C#

```
IndexQuery<T> FindBetween(
   object min,
   bool minInclusive,
   object max,
   bool maxInclusive,
   Order order
)
```

Parameters

min

Minimum key value to search for.

minInclusive

If **true**, the result includes items with the minimum key value.

max

Maximum key value to search for.

maxInclusive

If **true**, the result includes items with the maximum key value.

order

Optionally specifies the order of the key values to sort the result (C1.LiveLinq.Order.**Unordered** if sorting is not required).

Return Value

An object enumerating all items with key values within the specified limits.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IIndexScanner<T> Interface
IIndexScanner<T> Members

FindGreater Method
Minimum key value to search for.

If **true**, the result includes items with the specified key value. Otherwise, the result only includes those with keys strictly greater than the specified value.

Optionally specifies the order of the key values to sort the result (C1.LiveLinq.Order.**Unordered** if sorting is not required).

Finds items with keys greater than the specified value.

Syntax

Visual Basic (Declaration)

```
Function FindGreater( _
   ByVal key As Object, _
   ByVal inclusive As Boolean, _
   ByVal order As Order _
) As IndexQuery(Of T)
```

C#

```
IndexQuery<T> FindGreater(
   object key,
   bool inclusive,
   Order order
)
```

Parameters

key

Minimum key value to search for.

inclusive

If **true**, the result includes items with the specified key value. Otherwise, the result only includes those with keys strictly greater than the specified value.

order

Optionally specifies the order of the key values to sort the result (C1.LiveLinq.Order.**Unordered** if sorting is not required).

Return Value

An object enumerating all items whose key values are greater than the specified value.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IIndexScanner<T> Interface
IIndexScanner<T> Members

FindKeys Method
The key values to search for.

Optionally specifies the order of the key values to sort the result (C1.LiveLinq.Order.**Unordered** if sorting is not required).

Finds items containing any of the specified key values.

Syntax

Visual Basic (Declaration)

`Function FindKeys( _`
`  ByVal keys As IEnumerable, _`
`  ByVal order As Order _`
`) As IndexQuery(Of T)`

C#

`IndexQuery<T> FindKeys(`
`  IEnumerable keys,`
`  Order order`
`)`

Parameters

keys

The key values to search for.

order

Optionally specifies the order of the key values to sort the result (C1.LiveLinq.Order.**Unordered** if sorting is not required).

Return Value

An object enumerating all items whose key values belong to the specified key value collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IIndexScanner<T> Interface
IIndexScanner<T> Members

FindLess Method
Maximum key value to search for.

If **true**, the result includes items with the specified key value. Otherwise, the result only includes those with keys strictly less than the specified value.

Optionally specifies the order of the key values to sort the result (C1.LiveLinq.Order.**Unordered** if sorting is not required).

Finds items with keys less than the specified value.

Syntax

Visual Basic (Declaration)

```
Function FindLess( _
   ByVal key As Object, _
   ByVal inclusive As Boolean, _
   ByVal order As Order _
) As IndexQuery(Of T)
```

C#

```
IndexQuery<T> FindLess(
   object key,
   bool inclusive,
   Order order
)
```

Parameters

key

Maximum key value to search for.

inclusive

If **true**, the result includes items with the specified key value. Otherwise, the result only includes those with keys strictly less than the specified value.

order

Optionally specifies the order of the key values to sort the result (C1.LiveLinq.Order.**Unordered** if sorting is not required).

Return Value

An object enumerating all items whose key values are less than the specified value.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IIndexScanner<T> Interface
IIndexScanner<T> Members

FindStartingWith Method

The string to search for as the beginning of key value strings.

An optional condition that found items must satisfy.

Optionally specifies the order of the key values to sort the result (C1.LiveLinq.Order.**Unordered** if sorting is not required).

Finds items with string key values starting with the specified string.

Syntax

Visual Basic (Declaration)	
`Function FindStartingWith( _`	

```
   ByVal value As String, _
   ByVal keyPredicate As Func(Of String,Boolean), _
   ByVal order As Order _
) As IndexQuery(Of T,String)
```

C#

```
IndexQuery<T,string> FindStartingWith(
   string value,
   Func<string,bool> keyPredicate,
   Order order
)
```

Parameters

value

The string to search for as the beginning of key value strings.

keyPredicate

An optional condition that found items must satisfy.

order

Optionally specifies the order of the key values to sort the result (C1.LiveLinq.Order.**Unordered** if sorting is not required).

Return Value

An object enumerating all items whose key values are strings that have a beginning matching the specified string and satisfy the optional condition.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IIndexScanner<T> Interface
IIndexScanner<T> Members

GroupJoin Method

Correlates the items of this indexed collection with the items of another indexed collection and groups the results by the item of this collection.

Overload List

Overload	Description
GroupJoin<T2,TResult>(IIndexScanner<T2>,Func<T,IEnumerable<T2>,TResult>)	Correlates the items of this indexed collection with the items of another indexed collection and groups the results by the item of this collection.
GroupJoin<T2,TResult>(IEnumerable<T2>,Func<T2,Object>,Func<IEnumerable<T>,T2,TResult>)	Correlates the items of this indexed collection with the items of another

	sequence and groups the results by the item of the second sequence.
GroupJoin<T2,TResult>(IEnumerable<T2>,Func<T2,Object>,Func<T,IEnumerable<T2>,TResult>)	Correlates the items of this indexed collection with the items of another sequence and groups the results by the item of this collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IIndexScanner<T> Interface
IIndexScanner<T> Members

GroupJoin<T2,TResult>(IIndexScanner<T2>,Func<T,IEnumerable<T2>,TResult>) Method
The type of the elements of the second collection.

The type of the result elements.

The second indexed collection to join to this collection.

A function to create a result element from an element from this collection and a collection of matching elements from the second collection.

Correlates the items of this indexed collection with the items of another indexed collection and groups the results by the item of this collection.

Syntax

Visual Basic (Declaration)

```
Overloads Function GroupJoin
    (Of T2,TResult)( _
   ByVal source As IIndexScanner(Of T2), _
   ByVal resultSelector As Func(Of T,IEnumerable(Of T2),TResult) _
) As IEnumerable(Of TResult)
```

C#

```
IEnumerable<TResult> GroupJoin<T2,TResult>(
   IIndexScanner<T2> source,
   Func<T,IEnumerable<T2>,TResult> resultSelector
)
```

Parameters

source

The second indexed collection to join to this collection.

resultSelector

A function to create a result element from an element from this collection and a collection of matching elements from the second collection.

Type Parameters

T2

The type of the elements of the second collection.

TResult

The type of the result elements.

Return Value

Enumeration of objects obtained by applying the result selector to group pairs, where each pair consists of an item of this collection and the corresponding enumeration of the items of the second collection joined to it.

Remarks

Matching of two elements is performed by matching their keys.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IIndexScanner<T> Interface
IIndexScanner<T> Members
Overload List

GroupJoin<T2,TResult>(IEnumerable<T2>,Func<T2,Object>,Func<IEnumerable<T>,T2,TResult>) Method
The type of the elements of the second sequence.

The type of the result elements.

The second sequence to join to this collection.

A function to extract from an item of the second sequence the value to match against this collection's key value.

A function to create a result element from an element of the second sequence and the collection of matching elements from this collection.

Correlates the items of this indexed collection with the items of another sequence and groups the results by the item of the second sequence.

Syntax

Visual Basic (Declaration)

```
Overloads Function GroupJoin
    (Of T2,TResult)( _
   ByVal source As IEnumerable(Of T2), _
   ByVal keySelector As Func(Of T2,Object), _
   ByVal resultSelector As Func(Of IEnumerable(Of T),T2,TResult) _
) As IEnumerable(Of TResult)
```

C#

```
IEnumerable<TResult> GroupJoin<T2,TResult>(
   IEnumerable<T2> source,
   Func<T2,object> keySelector,
   Func<IEnumerable<T>,T2,TResult> resultSelector
)
```

Parameters

source

The second sequence to join to this collection.

keySelector

A function to extract from an item of the second sequence the value to match against this collection's key value.

resultSelector

A function to create a result element from an element of the second sequence and the collection of matching elements from this collection.

Type Parameters

T2

The type of the elements of the second sequence.

TResult

The type of the result elements.

Return Value

Enumeration of objects obtained by applying the result selector to group pairs, where each pair consists of an item of the second collection and the corresponding enumeration of the items of this collection joined to it.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IIndexScanner<T> Interface
IIndexScanner<T> Members
Overload List

GroupJoin<T2,TResult>(IEnumerable<T2>,Func<T2,Object>,Func<T,IEnumerable<T2>,TResult>) Method
The type of the elements of the second sequence.

The type of the result elements.

The second sequence to join to this collection.

A function to extract from an item of the second sequence the value to match against this collection's key value.

A function to create a result element from an element from this collection and a collection of matching elements from the second sequence.

Correlates the items of this indexed collection with the items of another sequence and groups the results by the item of this collection.

Syntax

Visual Basic (Declaration)
`Overloads Function GroupJoin`

```
    (Of T2,TResult)( _
   ByVal source As IEnumerable(Of T2), _
   ByVal keySelector As Func(Of T2,Object), _
   ByVal resultSelector As Func(Of T,IEnumerable(Of T2),TResult) _
) As IEnumerable(Of TResult)
```

C#

```
IEnumerable<TResult> GroupJoin<T2,TResult>(
   IEnumerable<T2> source,
   Func<T2,object> keySelector,
   Func<T,IEnumerable<T2>,TResult> resultSelector
)
```

Parameters

source

The second sequence to join to this collection.

keySelector

A function to extract from an item of the second sequence the value to match against this collection's key value.

resultSelector

A function to create a result element from an element from this collection and a collection of matching elements from the second sequence.

Type Parameters

T2

The type of the elements of the second sequence.

TResult

The type of the result elements.

Return Value

Enumeration of objects obtained by applying the result selector to group pairs, where each pair consists of an item of this collection and the corresponding enumeration of the items of the second sequence joined to it.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IIndexScanner<T> Interface
IIndexScanner<T> Members
Overload List

Join Method

Correlates the items of this indexed collection with the items of another sequence and returns the combined items with matching keys.

Overload List

Overload	Description
Join<T2,TResult>(IEnumerable<T2>,Func<T2,Object>,Func<T,T2,TResult>,JoinOperator)	Correlates the items of this indexed collection with the items of another sequence and returns the combined items with matching

	keys.
Join<T2,TResult>(IIndexScanner<T2>,Func<T,T2,TResult>,JoinOperator)	Correlates the items of this indexed collection with the items of another indexed collection and returns the combined items with matching keys.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IIndexScanner<T> Interface
IIndexScanner<T> Members

Join<T2,TResult>(IEnumerable<T2>,Func<T2,Object>,Func<T,T2,TResult>,JoinOperator) Method
The type of the elements of the second sequence.

The type of the result elements.

The second sequence to join to this collection.

A function to extract from an item of the second sequence the value to match against this collection's key value.

A function to create a result element from two matching elements.

A comparison operator to match elements.

Correlates the items of this indexed collection with the items of another sequence and returns the combined items with matching keys.

Syntax

Visual Basic (Declaration)

```
Overloads Function Join
    (Of T2,TResult)( _
   ByVal source As IEnumerable(Of T2), _
   ByVal keySelector As Func(Of T2,Object), _
   ByVal resultSelector As Func(Of T,T2,TResult), _
   ByVal op As JoinOperator _
) As IEnumerable(Of TResult)
```

C#

```
IEnumerable<TResult> Join<T2,TResult>(
   IEnumerable<T2> source,
   Func<T2,object> keySelector,
   Func<T,T2,TResult> resultSelector,
   JoinOperator op
)
```

Parameters

source

The second sequence to join to this collection.

keySelector

A function to extract from an item of the second sequence the value to match against this collection's key value.

resultSelector

A function to create a result element from two matching elements.

op

A comparison operator to match elements.

Type Parameters

T2

The type of the elements of the second sequence.

TResult

The type of the result elements.

Return Value

Enumeration of objects obtained by applying the result selector to pairs of joined elements of the two collections.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IIndexScanner<T> Interface
IIndexScanner<T> Members
Overload List

Join<T2,TResult>(IIndexScanner<T2>,Func<T,T2,TResult>,JoinOperator) Method
The type of the elements of the second collection.

The type of the result elements.

The second indexed collection to join to this collection.

A function to create a result element from two matching elements.

A comparison operator to match elements.

Correlates the items of this indexed collection with the items of another indexed collection and returns the combined items with matching keys.

Syntax

Visual Basic (Declaration)

```
Overloads Function Join
    (Of T2,TResult)( _
   ByVal source As IIndexScanner(Of T2), _
   ByVal resultSelector As Func(Of T,T2,TResult), _
   ByVal op As JoinOperator _
) As IEnumerable(Of TResult)
```

C#

```
IEnumerable<TResult> Join<T2,TResult>(
   IIndexScanner<T2> source,
   Func<T,T2,TResult> resultSelector,
   JoinOperator op
)
```

Parameters

source

The second indexed collection to join to this collection.

resultSelector

A function to create a result element from two matching elements.

op

A comparison operator to match elements.

Type Parameters

T2

The type of the elements of the second collection.

TResult

The type of the result elements.

Return Value

Enumeration of objects obtained by applying the result selector to pairs of joined elements of the two collections.

Remarks

Matching of two elements is performed by matching their keys.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IIndexScanner<T> Interface
IIndexScanner<T> Members
Overload List

Properties
For a list of all members of this type, see IIndexScanner<T> members.

Public Properties

	Name	Description
	Definition	Gets an C1.LiveLinq.Indexing.Index<T> or a C1.LiveLinq.Indexing.Subindex<T> definition on which the scanner is based.
	KeyCount	Gets the number of distinct key values in all items of this collection.
	ParentScanner	Gets an index or a subindex scanner that is the parent of a subindex scanner.

Top

See Also

Reference

IIndexScanner<T> Interface
C1.LiveLinq.Indexing.Search Namespace

Definition Property
Gets an C1.LiveLinq.Indexing.Index<T> or a C1.LiveLinq.Indexing.Subindex<T> definition on which the scanner is based.

Syntax

Visual Basic (Declaration)
`ReadOnly Property Definition As IndexDefinition(Of T)`
C#
`IndexDefinition<T> Definition {get;}`

Property Value

For a subindex scanner, returns a C1.LiveLinq.Indexing.Subindex<T>. For an index scanner, returns the same C1.LiveLinq.Indexing.Index<T> object as the scanner itself.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IIndexScanner<T> Interface
IIndexScanner<T> Members

KeyCount Property
Gets the number of distinct key values in all items of this collection.

Syntax

Visual Basic (Declaration)
`ReadOnly Property KeyCount As Integer`

C#

```
int KeyCount {get;}
```

Property Value

Number of distinct key values in the collection.

Remarks

This number is not the same as the number of elements in the collection, unless the index key is a unique key of that collection, see C1.LiveLinq.Indexing.IndexDefinition<T>.KeyIsUnique.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IIndexScanner<T> Interface
IIndexScanner<T> Members

ParentScanner Property
Gets an index or a subindex scanner that is the parent of a subindex scanner.

Syntax

Visual Basic (Declaration)

```
ReadOnly Property ParentScanner As IIndexScanner(Of T)
```

C#

```
IIndexScanner<T> ParentScanner {get;}
```

Property Value

Parent scanner for a subindex scanner; **null** for an index scanner.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

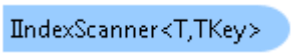
IIndexScanner<T> Interface
IIndexScanner<T> Members

IIndexScanner<T,TKey>
The type of the elements of the indexed collection.

The type of the index key.

Represents an index or a subindex in its capacity of scanning through data. Provides methods for searching data items.

Object Model



Syntax

Visual Basic (Declaration)

`Public Interface IIndexScanner`
`  (Of T,TKey)`
`  Inherits IIndexScanner(Of T)`

C#
`public interface IIndexScanner<T,TKey> : IIndexScanner<T>`

Type Parameters

T

The type of the elements of the indexed collection.

TKey

The type of the index key.

Remarks

This interface is implemented by C1.LiveLinq.Indexing.Index<T,TKey>. It is also used by subindexes, but there it is not directly implemented by C1.LiveLinq.Indexing.Subindex<T,TKey>, but rather returned by the IndexQuery<T>.Subindex method because it depends on the item found by an index or a subindex that is the parent of that subindex.

IIndexScanner<T,TKey> has an untyped key counterpart IIndexScanner<T> that is used with untyped key classes C1.LiveLinq.Indexing.Index<T> and C1.LiveLinq.Indexing.Subindex<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

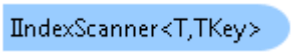
IIndexScanner<T,TKey> Members
C1.LiveLinq.Indexing.Search Namespace

Overview
The type of the elements of the indexed collection.

The type of the index key.

Represents an index or a subindex in its capacity of scanning through data. Provides methods for searching data items.

Object Model



Syntax

Visual Basic (Declaration)	
`Public Interface IIndexScanner` `    (Of T,TKey)`	

```
Inherits IIndexScanner(Of T)
```

C#

```
public interface IIndexScanner<T,TKey> : IIndexScanner<T>
```

Type Parameters

T

The type of the elements of the indexed collection.

TKey

The type of the index key.

Remarks

This interface is implemented by C1.LiveLinq.Indexing.Index<T,TKey>. It is also used by subindexes, but there it is not directly implemented by C1.LiveLinq.Indexing.Subindex<T,TKey>, but rather returned by the IndexQuery<T>.Subindex method because it depends on the item found by an index or a subindex that is the parent of that subindex.

IIndexScanner<T,TKey> has an untyped key counterpart IIndexScanner<T> that is used with untyped key classes C1.LiveLinq.Indexing.Index<T> and C1.LiveLinq.Indexing.Subindex<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IIndexScanner<T,TKey> Members
C1.LiveLinq.Indexing.Search Namespace

Members
Methods

The following tables list the members exposed by IIndexScanner<T,TKey>.

Public Methods

	Name	Description
	All	Gets all items in the indexed collection.
	ContainsKey	Returns a value that indicates whether the collection contains an item with the given key value.
	Find	Finds items with the specified key value.
	FindBetween	Finds items with key values in the interval between the specified values.
	FindGreater	Finds items with keys greater than the specified value.
	FindKeys	Finds items containing any of the specified key values.
	FindLess	Finds items with keys less than the specified value.
	GroupJoin	Overloaded. Correlates the items of this indexed collection with the items of another indexed collection and groups the results by the item of this collection.
	Join	Overloaded. Correlates the items of this indexed collection with the items of another sequence and returns the combined items with matching keys.
	Keys	Gets distinct key values in all items of this collection.

Top

See Also

Reference

IIndexScanner<T,TKey> Interface
C1.LiveLinq.Indexing.Search Namespace

Methods
For a list of all members of this type, see IIndexScanner<T,TKey> members.

Public Methods

	Name	Description
	All	Gets all items in the indexed collection.
	ContainsKey	Returns a value that indicates whether the collection contains an item with the given key value.
	Find	Finds items with the specified key value.
	FindBetween	Finds items with key values in the interval between the specified values.
	FindGreater	Finds items with keys greater than the specified value.
	FindKeys	Finds items containing any of the specified key values.
	FindLess	Finds items with keys less than the specified value.
	GroupJoin	Overloaded. Correlates the items of this indexed collection with the items of another indexed collection and groups the results by the item of this collection.
	Join	Overloaded. Correlates the items of this indexed collection with the items of another sequence and returns the combined items with matching keys.
	Keys	Gets distinct key values in all items of this collection.

Top

See Also

Reference

IIndexScanner<T,TKey> Interface
C1.LiveLinq.Indexing.Search Namespace

All Method
Specifies the order of the key values to sort the result.

Gets all items in the indexed collection.

Syntax

Visual Basic (Declaration)

`Function All( _`
`  ByVal order As Order _`
`) As IndexQuery(Of T,TKey)`

C#

`IndexQuery<T,TKey> All(`
`  Order order`
`)`

Parameters

order

Specifies the order of the key values to sort the result.

Return Value

All items of the collection in the specified order of their key values.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IIndexScanner<T,TKey> Interface
IIndexScanner<T,TKey> Members

ContainsKey Method
The key value to search for

Returns a value that indicates whether the collection contains an item with the given key value.

Syntax

Visual Basic (Declaration)

```
Function ContainsKey( _
   ByVal key As TKey _
) As Boolean
```

C#

```
bool ContainsKey(
   TKey key
)
```

Parameters

key

The key value to search for

Return Value

true if the collection contains an element with the specified key value; otherwise, **false**.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IIndexScanner<T,TKey> Interface
IIndexScanner<T,TKey> Members

Find Method
The key value to search for.

Finds items with the specified key value.

Syntax

Visual Basic (Declaration)

```
Function Find( _
```

```
   ByVal key As TKey _
) As IndexQuery(Of T,TKey)
```

C#

```
IndexQuery<T,TKey> Find(
   TKey key
)
```

Parameters

key

The key value to search for.

Return Value

An object enumerating items having the specified key value.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IIndexScanner<T,TKey> Interface
IIndexScanner<T,TKey> Members

FindBetween Method

Minimum key value to search for.

If **true**, the result includes items with the minimum key value.

Maximum key value to search for.

If **true**, the result includes items with the maximum key value.

An optional condition that found items must satisfy.

Optionally specifies the order of the key values to sort the result (C1.LiveLinq.Order.**Unordered** if sorting is not required).

Finds items with key values in the interval between the specified values.

Syntax

Visual Basic (Declaration)

```
Function FindBetween( _
   ByVal min As TKey, _
   ByVal minInclusive As Boolean, _
   ByVal max As TKey, _
   ByVal maxInclusive As Boolean, _
   ByVal keyPredicate As Func(Of TKey,Boolean), _
   ByVal order As Order _
) As IndexQuery(Of T,TKey)
```

C#

```
IndexQuery<T,TKey> FindBetween(
   TKey min,
   bool minInclusive,
   TKey max,
   bool maxInclusive,
   Func<TKey,bool> keyPredicate,
   Order order
)
```

Parameters

min

Minimum key value to search for.

minInclusive

If **true**, the result includes items with the minimum key value.

max

Maximum key value to search for.

maxInclusive

If **true**, the result includes items with the maximum key value.

keyPredicate

An optional condition that found items must satisfy.

order

Optionally specifies the order of the key values to sort the result (C1.LiveLinq.Order.**Unordered** if sorting is not required).

Return Value

An object enumerating all items with key values within the specified limits.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IIndexScanner<T,TKey> Interface
IIndexScanner<T,TKey> Members

FindGreater Method

Minimum key value to search for.

If **true**, the result includes items with the specified key value. Otherwise, the result only includes those with keys strictly greater than the specified value.

An optional condition that found items must satisfy.

Optionally specifies the order of the key values to sort the result (C1.LiveLinq.Order.**Unordered** if sorting is not required).

Finds items with keys greater than the specified value.

Syntax

Visual Basic (Declaration)

```
Function FindGreater( _
   ByVal key As TKey, _
   ByVal inclusive As Boolean, _
   ByVal keyPredicate As Func(Of TKey,Boolean), _
```

```
   ByVal order As Order _
) As IndexQuery(Of T,TKey)
```

C#

```
IndexQuery<T,TKey> FindGreater(
   TKey key,
   bool inclusive,
   Func<TKey,bool> keyPredicate,
   Order order
)
```

Parameters

key

Minimum key value to search for.

inclusive

If **true**, the result includes items with the specified key value. Otherwise, the result only includes those with keys strictly greater than the specified value.

keyPredicate

An optional condition that found items must satisfy.

order

Optionally specifies the order of the key values to sort the result (C1.LiveLinq.Order.**Unordered** if sorting is not required).

Return Value

An object enumerating all items whose key values are greater than the specified value.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IIndexScanner<T,TKey> Interface
IIndexScanner<T,TKey> Members

FindKeys Method

The key values to search for.

Optionally specifies the order of the key values to sort the result (C1.LiveLinq.Order.**Unordered** if sorting is not required).

Finds items containing any of the specified key values.

Syntax

Visual Basic (Declaration)

`Function FindKeys( _`
`  ByVal keys As IEnumerable(Of TKey), _`
`  ByVal order As Order _`
`) As IndexQuery(Of T,TKey)`

C#

`IndexQuery<T,TKey> FindKeys(`
`  IEnumerable<TKey> keys,`
`  Order order`
`)`

Parameters

keys

The key values to search for.

order

Optionally specifies the order of the key values to sort the result (C1.LiveLinq.Order.**Unordered** if sorting is not required).

Return Value

An object enumerating all items whose key values belong to the specified key value collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IIndexScanner<T,TKey> Interface
IIndexScanner<T,TKey> Members

FindLess Method
Maximum key value to search for.

If **true**, the result includes items with the specified key value. Otherwise, the result only includes those with keys strictly less than the specified value.

An optional condition that found items must satisfy.

Optionally specifies the order of the key values to sort the result (C1.LiveLinq.Order.**Unordered** if sorting is not required).

Finds items with keys less than the specified value.

Syntax

Visual Basic (Declaration)

```
Function FindLess( _
   ByVal key As TKey, _
   ByVal inclusive As Boolean, _
   ByVal keyPredicate As Func(Of TKey,Boolean), _
   ByVal order As Order _
) As IndexQuery(Of T,TKey)
```

C#

```
IndexQuery<T,TKey> FindLess(
   TKey key,
   bool inclusive,
   Func<TKey,bool> keyPredicate,
   Order order
)
```

Parameters

key

Maximum key value to search for.

inclusive

If **true**, the result includes items with the specified key value. Otherwise, the result only includes those with keys strictly less than the specified value.

keyPredicate

An optional condition that found items must satisfy.

order

Optionally specifies the order of the key values to sort the result (C1.LiveLinq.Order.**Unordered** if sorting is not required).

Return Value

An object enumerating all items whose key values are less than the specified value.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IIndexScanner<T,TKey> Interface
IIndexScanner<T,TKey> Members

GroupJoin Method

Correlates the items of this indexed collection with the items of another indexed collection and groups the results by the item of this collection.

Overload List

Overload	Description

GroupJoin<T2,TResult>(IIndexScanner<T2,TKey>,Func<T,IEnumerable<T2>,TResult>)	Correlates the items of this indexed collection with the items of another indexed collection and groups the results by the item of this collection.
GroupJoin<T2,TResult>(IEnumerable<T2>,Func<T2,TKey>,Func<IEnumerable<T>,T2,TResult>)	Correlates the items of this indexed collection with the items of another sequence and groups the results by the item of the

	second sequence.
GroupJoin<T2,TResult>(IEnumerable<T2>,Func<T2,TKey>,Func<T,IEnumerable<T2>,TResult>)	Correlates the items of this indexed collection with the items of another sequence and groups the results by the item of this collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IIndexScanner<T,TKey> Interface
IIndexScanner<T,TKey> Members

GroupJoin<T2,TResult>(IIndexScanner<T2,TKey>,Func<T,IEnumerable<T2>,TResult>) Method
The type of the elements of the second collection.

The type of the result elements.

The second indexed collection to join to this collection.

A function to create a result element from an element from this collection and a collection of matching elements from the second collection.

Correlates the items of this indexed collection with the items of another indexed collection and groups the results by the item of this collection.

Syntax

Visual Basic (Declaration)

```
Overloads Function GroupJoin
    (Of T2,TResult)( _
   ByVal source As IIndexScanner(Of T2,TKey), _
   ByVal resultSelector As Func(Of T,IEnumerable(Of T2),TResult) _
) As IEnumerable(Of TResult)
```

C#

```
IEnumerable<TResult> GroupJoin<T2,TResult>(
   IIndexScanner<T2,TKey> source,
   Func<T,IEnumerable<T2>,TResult> resultSelector
)
```

Parameters

source

The second indexed collection to join to this collection.

resultSelector

A function to create a result element from an element from this collection and a collection of matching elements from the second collection.

Type Parameters

T2

The type of the elements of the second collection.

TResult

The type of the result elements.

Return Value

Enumeration of objects obtained by applying the result selector to group pairs, where each pair consists of an item of this collection and the corresponding enumeration of the items of the second collection joined to it.

Remarks

Matching of two elements is performed by matching their keys.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IIndexScanner<T,TKey> Interface
IIndexScanner<T,TKey> Members
Overload List

GroupJoin<T2,TResult>(IEnumerable<T2>,Func<T2,TKey>,Func<IEnumerable<T>,T2,TResult>) Method
The type of the elements of the second sequence.

The type of the result elements.

The second sequence to join to this collection.

A function to extract from an item of the second sequence the value to match against this collection's key value.

A function to create a result element from an element of the second sequence and the collection of matching elements from this collection.

Correlates the items of this indexed collection with the items of another sequence and groups the results by the item of the second sequence.

Syntax

Visual Basic (Declaration)	
`Overloads Function GroupJoin` `    (Of T2,TResult)( _`	

```
   ByVal source As IEnumerable(Of T2), _
   ByVal keySelector As Func(Of T2,TKey), _
   ByVal resultSelector As Func(Of IEnumerable(Of T),T2,TResult) _
) As IEnumerable(Of TResult)
```

C#

```
IEnumerable<TResult> GroupJoin<T2,TResult>(
   IEnumerable<T2> source,
   Func<T2,TKey> keySelector,
   Func<IEnumerable<T>,T2,TResult> resultSelector
)
```

Parameters

source

The second sequence to join to this collection.

keySelector

A function to extract from an item of the second sequence the value to match against this collection's key value.

resultSelector

A function to create a result element from an element of the second sequence and the collection of matching elements from this collection.

Type Parameters

T2

The type of the elements of the second sequence.

TResult

The type of the result elements.

Return Value

Enumeration of objects obtained by applying the result selector to group pairs, where each pair consists of an item of the second collection and the corresponding enumeration of the items of this collection joined to it.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IIndexScanner<T,TKey> Interface
IIndexScanner<T,TKey> Members
Overload List

GroupJoin<T2,TResult>(IEnumerable<T2>,Func<T2,TKey>,Func<T,IEnumerable<T2>,TResult>) Method
The type of the elements of the second sequence.

The type of the result elements.

The second sequence to join to this collection.

A function to extract from an item of the second sequence the value to match against this collection's key value.

A function to create a result element from an element from this collection and a collection of matching elements from the second sequence.

Correlates the items of this indexed collection with the items of another sequence and groups the results by the item of this collection.

Syntax

Visual Basic (Declaration)

```
Overloads Function GroupJoin
    (Of T2,TResult)( _
   ByVal source As IEnumerable(Of T2), _
   ByVal keySelector As Func(Of T2,TKey), _
   ByVal resultSelector As Func(Of T,IEnumerable(Of T2),TResult) _
) As IEnumerable(Of TResult)
```

C#

```
IEnumerable<TResult> GroupJoin<T2,TResult>(
   IEnumerable<T2> source,
```

```
    Func<T2,TKey> keySelector,
    Func<T,IEnumerable<T2>,TResult> resultSelector
)
```

Parameters

source

The second sequence to join to this collection.

keySelector

A function to extract from an item of the second sequence the value to match against this collection's key value.

resultSelector

A function to create a result element from an element from this collection and a collection of matching elements from the second sequence.

Type Parameters

T2

The type of the elements of the second sequence.

TResult

The type of the result elements.

Return Value

Enumeration of objects obtained by applying the result selector to group pairs, where each pair consists of an item of this collection and the corresponding enumeration of the items of the second sequence joined to it.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IIndexScanner<T,TKey> Interface
IIndexScanner<T,TKey> Members
Overload List

Join Method

Correlates the items of this indexed collection with the items of another sequence and returns the combined items with matching keys.

Overload List

Overload	Description
Join<T2,TResult>(IEnumerable<T2>,Func<T2,TKey>,Func<T,T2,TResult>,JoinOperator)	Correlates the items of this indexed collection with the items of another sequence and returns the combined items with matching keys.
Join<T2,TResult>(IIndexScanner<T2,TKey>,Func<T,T2,TResult>,JoinOperator)	Correlates the items of this indexed collection with the items of another indexed

	collection and returns the combined items with matching keys.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IIndexScanner<T,TKey> Interface
IIndexScanner<T,TKey> Members

Join<T2,TResult>(IEnumerable<T2>,Func<T2,TKey>,Func<T,T2,TResult>,JoinOperator) Method
The type of the elements of the second sequence.

The type of the result elements.

The second sequence to join to this collection.

A function to extract from an item of the second sequence the value to match against this collection's key value.

A function to create a result element from two matching elements.

A comparison operator to match elements.

Correlates the items of this indexed collection with the items of another sequence and returns the combined items with matching keys.

Syntax

Visual Basic (Declaration)
`Overloads Function Join`

```
    (Of T2,TResult)( _
   ByVal source As IEnumerable(Of T2), _
   ByVal keySelector As Func(Of T2,TKey), _
   ByVal resultSelector As Func(Of T,T2,TResult), _
   ByVal op As JoinOperator _
) As IEnumerable(Of TResult)
```

C#

```
IEnumerable<TResult> Join<T2,TResult>(
   IEnumerable<T2> source,
   Func<T2,TKey> keySelector,
   Func<T,T2,TResult> resultSelector,
   JoinOperator op
)
```

Parameters

source

The second sequence to join to this collection.

keySelector

A function to extract from an item of the second sequence the value to match against this collection's key value.

resultSelector

A function to create a result element from two matching elements.

op

A comparison operator to match elements.

Type Parameters

T2

The type of the elements of the second sequence.

TResult

The type of the result elements.

Return Value

Enumeration of objects obtained by applying the result selector to pairs of joined elements of the two collections.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IIndexScanner<T,TKey> Interface
IIndexScanner<T,TKey> Members
Overload List

Join<T2,TResult>(IIndexScanner<T2,TKey>,Func<T,T2,TResult>,JoinOperator) Method
The type of the elements of the second collection.

The type of the result elements.

The second indexed collection to join to this collection.

A function to create a result element from two matching elements.

A comparison operator to match elements.

Correlates the items of this indexed collection with the items of another indexed collection and returns the combined items with matching keys.

Syntax

Visual Basic (Declaration)

```
Overloads Function Join
    (Of T2,TResult)( _
   ByVal source As IIndexScanner(Of T2,TKey), _
   ByVal resultSelector As Func(Of T,T2,TResult), _
   ByVal op As JoinOperator _
) As IEnumerable(Of TResult)
```

C#

```
IEnumerable<TResult> Join<T2,TResult>(
   IIndexScanner<T2,TKey> source,
   Func<T,T2,TResult> resultSelector,
   JoinOperator op
)
```

Parameters

source

The second indexed collection to join to this collection.

resultSelector

A function to create a result element from two matching elements.

op

A comparison operator to match elements.

Type Parameters

T2

The type of the elements of the second collection.

TResult

The type of the result elements.

Return Value

Enumeration of objects obtained by applying the result selector to pairs of joined elements of the two collections.

Remarks

Matching of two elements is performed by matching their keys.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IIndexScanner<T,TKey> Interface
IIndexScanner<T,TKey> Members
Overload List

Keys Method

Specifies the order of the key values to sort the result.

Gets distinct key values in all items of this collection.

Syntax

Visual Basic (Declaration)

`Function Keys( _`
`  ByVal order As Order _`
`) As IEnumerable(Of TKey)`

C#

`IEnumerable<TKey> Keys(`
`  Order order`
`)`

Parameters

order

Specifies the order of the key values to sort the result.

Return Value

All distinct key values contained in the items of the collection in the specified order.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IIndexScanner<T,TKey> Interface
IIndexScanner<T,TKey> Members

C1.LiveLinq.Listeners Namespace

Overview

Classes

	Class	Description
	PropertyChangeListener<T>	Represents a listener object used by LiveLinq to receive notifications of changes to property values in a collection element object. Represents a listener object used by LiveLinq to receive notifications of changes to property values in a collection element object.

See Also

Reference

C1.LiveLinq.4 Assembly

Classes

PropertyChangeListener<T>
The type of the elements in the collection.

Represents a listener object used by LiveLinq to receive notifications of changes to property values in a collection element object. Represents a listener object used by LiveLinq to receive notifications of changes to property values in a collection element object.

Object Model

Syntax

Visual Basic (Declaration)
`Public MustInherit Class PropertyChangeListener(Of T)`

C#

```
public abstract class PropertyChangeListener<T>
```

Type Parameters

T

The type of the elements in the collection.

Remarks

In most cases, the default listener mechanism performs its function and does not need user intervention. As an advanced functionality, LiveLinq allows the user to customize default listeners. It may be needed in those rare cases when you use C1.LiveLinq.Collections.IndexedCollection<T> with element class **T** that does not provide property change notifications, and you can't add such notifications, nor by deriving the class from C1.LiveLinq.Collections.IndexableObject nor by other means, but you have some other way of knowing when property changes occur (maybe you can listen to some events, for example).

Then you can define your own class derived from the base class **PropertyChangeListener<T>** and pass an object of that class to the C1.LiveLinq.Collections.IndexedCollection<T> constructor.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.Listeners.PropertyChangeListener<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

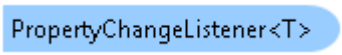
PropertyChangeListener<T> Members
C1.LiveLinq.Listeners Namespace

Overview

The type of the elements in the collection.

Represents a listener object used by LiveLinq to receive notifications of changes to property values in a collection element object. Represents a listener object used by LiveLinq to receive notifications of changes to property values in a collection element object.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit Class PropertyChangeListener(Of T)`
C#
`public abstract class PropertyChangeListener<T>`

Type Parameters

T

The type of the elements in the collection.

Remarks

In most cases, the default listener mechanism performs its function and does not need user intervention. As an advanced functionality, LiveLinq allows the user to customize default listeners. It may be needed in those rare cases when you use C1.LiveLinq.Collections.IndexedCollection<T> with element class **T** that does not provide property change notifications, and you can't add such notifications, nor by deriving the class from C1.LiveLinq.Collections.IndexableObject nor by other means, but you have some other way of knowing when property changes occur (maybe you can listen to some events, for example).

Then you can define your own class derived from the base class **PropertyChangeListener<T>** and pass an object of that class to the C1.LiveLinq.Collections.IndexedCollection<T> constructor.

Inheritance Hierarchy

System.Object
C1.LiveLinq.Listeners.PropertyChangeListener<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

PropertyChangeListener<T> Members
C1.LiveLinq.Listeners Namespace

Members
Methods Events

The following tables list the members exposed by PropertyChangeListener<T>.

Public Methods

	Name	Description
	Clear	Stop listening to all, don't listen to any objects.
S	CreateDefault	Creates the default listener used by LiveLinq to listen to property change notifications in objects of type **T**.
	GetListeningProperties	Gets the list of property names for which change notifications are supported.
	StartListening	Start listening to property changes in a particular object.
	StopListening	Stop listening to property changes in a particular object.

Top

Public Events

	Name	Description
	PropertyChanged	Occurs after a property has changed its value.

Top

See Also

Reference

PropertyChangeListener<T> Class
C1.LiveLinq.Listeners Namespace

Methods
For a list of all members of this type, see PropertyChangeListener<T> members.

Public Methods

	Name	Description
	Clear	Stop listening to all, don't listen to any objects.
S	CreateDefault	Creates the default listener used by LiveLinq to listen to property change notifications in objects of type **T**.
	GetListeningProperties	Gets the list of property names for which change notifications are supported.
	StartListening	Start listening to property changes in a particular object.
	StopListening	Stop listening to property changes in a particular object.

Top

See Also

Reference

PropertyChangeListener<T> Class
C1.LiveLinq.Listeners Namespace

Clear Method

Stop listening to all, don't listen to any objects.

Syntax

Visual Basic (Declaration)
`Public MustOverride Sub Clear()`
C#
`public abstract void Clear()`

Remarks

LiveLinq calls this method when a collection is cleared, all objects are removed from it, so it no longer needs to be notified of changes in any of those objects.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

PropertyChangeListener<T> Class
PropertyChangeListener<T> Members

CreateDefault Method

Creates the default listener used by LiveLinq to listen to property change notifications in objects of type **T**.

Syntax

Visual Basic (Declaration)
`Public Shared Function CreateDefault() As PropertyChangeListener(Of T)`
C#
`public static PropertyChangeListener<T> CreateDefault()`

Return Value

The default listener handling property change notifications.

Remarks

For ADO.NET and XML objects (**DataRow**, **DataRowView**, **XContainer**), their corresponding property change notifications are used.

For objects implementing **INotifyPropertyChanged** interface, that interface is used.

If the class is neither of the above, the default listener tries to use two change notification patterns if they are present in the class:

- events named *propertyName*Changed
- dependency properties of WPF classes

These patterns don't have to exist in the class, it's just a last ditch attempt of the default listener to find a notification mechanism if the standard one is not found.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

PropertyChangeListener<T> Class
PropertyChangeListener<T> Members

GetListeningProperties Method
Gets the list of property names for which change notifications are supported.

Syntax

Visual Basic (Declaration)
`Public MustOverride Function GetListeningProperties() As IEnumerable(Of String)`
C#
`public abstract IEnumerable<string> GetListeningProperties()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

PropertyChangeListener<T> Class
PropertyChangeListener<T> Members

StartListening Method
The object to listen to.

Start listening to property changes in a particular object.

Syntax

Visual Basic (Declaration)

`Public MustOverride Sub StartListening( _`
`  ByVal item As T _`
`)`

C#

`public abstract void StartListening(`
`  T item`
`)`

Parameters

item

The object to listen to.

Remarks

LiveLinq calls this method when a new object is added to a collection, so LiveLinq needs to be notified of changes to property values in that object.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

PropertyChangeListener<T> Class
PropertyChangeListener<T> Members

StopListening Method
The object to stop listening to.

Stop listening to property changes in a particular object.

Syntax

Visual Basic (Declaration)

```
Public MustOverride Sub StopListening( _
   ByVal item As T _
)
```

C#

```
public abstract void StopListening(
   T item
)
```

Parameters

item

The object to stop listening to.

Remarks

LiveLinq calls this method when an object is removed from a collection, so LiveLinq no longer needs to be notified of changes to property values in that object.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

PropertyChangeListener<T> Class
PropertyChangeListener<T> Members

Events
For a list of all members of this type, see PropertyChangeListener<T> members.

Public Events

	Name	Description
	PropertyChanged	Occurs after a property has changed its value.

Top

See Also

Reference

PropertyChangeListener<T> Class
C1.LiveLinq.Listeners Namespace

PropertyChanged Event
Occurs after a property has changed its value.

Syntax

Visual Basic (Declaration)
`Public Event PropertyChanged As PropertyChangedEventHandler`
C#
`public event PropertyChangedEventHandler PropertyChanged`

Event Data

The event handler receives an argument of type PropertyChangedEventArgs containing data related to this event. The following **PropertyChangedEventArgs** properties provide information specific to this event.

Property	Description
PropertyName	

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

PropertyChangeListener<T> Class
PropertyChangeListener<T> Members

C1.LiveLinq.LiveViews Namespace

Overview

Classes

	Class	Description
	AggregationView<TSource,TResult>	Represents a view having a single element calculated by aggregating a source view.
	GroupView<TKey,TElement>	A group in a grouping view.
	OrderedView<T>	Represents a sorted view.
	PropertyIsNotVirtualException	Represents an exception that indicates that a property used in a result selector of a live view is not virtual.
	View	Base class for the View<T> class. Contains members that

		don't depend on the element type *T*.
	View<T>	Represents a *live view*: a LINQ query result that supports two-way data binding and is kept up-to-date with base data.
	ViewRow	Represents a view element (item) for the purposes of dynamic, programmatic access to its properties and data binding.
	ViewRowAddingEventArgs	Provides data for the ViewRowCollection.ViewRowAdding event.
	ViewRowCollection	Represents a collection of ViewRow objects used for programmatic access to view elements (items) and for data binding.
	ViewRowPropertyInfo	Allows to control certain behavior of a property of the element type of a View.

Enumerations

	Enumeration	Description
	DataBindingMode	Enumeration of the possible data binding modes. It is used by the View.DataBindingMode property.
	ViewMaintenanceMode	Specifies how a view is synchronized with changes in its base data.
	ViewOrder	Specifies whether and how a view must preserve item order if it exists in the source.
	ViewRowState	The state of a view row with regard to edit, add and delete operations if they are performed directly on the view.

See Also

Reference

C1.LiveLinq.4 Assembly

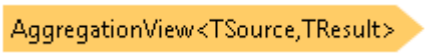
Classes

AggregationView<TSource,TResult>
The type of the elements of the source view.

The type of the single element of the aggregation view.

Represents a view having a single element calculated by aggregating a source view.

Object Model



Syntax

Visual Basic (Declaration)

`Public Class AggregationView`
`  (Of TSource,TResult)`
`  Inherits View(Of TResult)`
`  Implements C1.LiveLinq.Indexing.IIndexedSource(Of TResult),`
`C1.LiveLinq.IObservableSource(Of TResult)`

C#

`public class AggregationView<TSource,TResult> : View<TResult>,`
`C1.LiveLinq.Indexing.IIndexedSource<TResult>,`
`C1.LiveLinq.IObservableSource<TResult>`

Type Parameters

TSource

The type of the elements of the source view.

TResult

The type of the single element of the aggregation view.

Inheritance Hierarchy

System.Object
C1.LiveLinq.LiveViews.View
C1.LiveLinq.LiveViews.View<T>
C1.LiveLinq.LiveViews.AggregationView<TSource,TResult>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

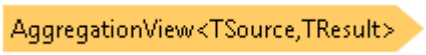
AggregationView<TSource,TResult> Members
C1.LiveLinq.LiveViews Namespace

Overview
The type of the elements of the source view.

The type of the single element of the aggregation view.

Represents a view having a single element calculated by aggregating a source view.

Object Model



Syntax

Visual Basic (Declaration)

```
Public Class AggregationView
    (Of TSource,TResult)
   Inherits View(Of TResult)
   Implements C1.LiveLinq.Indexing.IIndexedSource(Of TResult),
C1.LiveLinq.IObservableSource(Of TResult)
```

C#

```
public class AggregationView<TSource,TResult> : View<TResult>,
C1.LiveLinq.Indexing.IIndexedSource<TResult>,
```

```
C1.LiveLinq.IObservableSource<TResult>
```

Type Parameters

TSource

The type of the elements of the source view.

TResult

The type of the single element of the aggregation view.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.View
    C1.LiveLinq.LiveViews.View<T>
      C1.LiveLinq.LiveViews.AggregationView<TSource,TResult>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

AggregationView<TSource,TResult> Members
C1.LiveLinq.LiveViews Namespace

Members
Properties Methods Events

The following tables list the members exposed by AggregationView<TSource,TResult>.

Public Properties

	Name	Description
	Count	Gets the total number of elements in the view. (Inherited from C1.LiveLinq.LiveViews.View)

	CurrentItem	Gets the current item in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	DataBindingMode	Gets or sets the data binding mode for this view. (Inherited from C1.LiveLinq.LiveViews.View)
	DeferredMaintenance	Gets the effective value of MaintenanceMode. (Inherited from C1.LiveLinq.LiveViews.View)
	Indexes	Gets the collection of indexes for this view. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)
	IsReadOnly	Gets a value indicating whether this view is read-only, not updatable. (Inherited from C1.LiveLinq.LiveViews.View)
	Item	Gets the view item (element) at the specified ordinal position. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)
	MaintenanceMode	Gets or sets a value controlling how the view is synchronized with changes in its base data. (Inherited from C1.LiveLinq.LiveViews.View)
	MoveToFirstOnReset	Gets or sets a value indicating that the first item must be made current after initial loading or reset (on any C1.LiveLinq.SourceChangeType.Reset notification) if current item was not set by other means. The default is True. (Inherited from C1.LiveLinq.LiveViews.View)
	Order	Gets a value indicating whether and how this view preserves item order if it exists in its base data source. (Inherited from C1.LiveLinq.LiveViews.View)
	Transaction	Gets an instance of C1.LiveLinq.ITransaction associated with the view. If a view has a transaction associated with it, that transaction's scope is opened automatically every time the view is updated, so the programmer does not need to do it manually in code. (Inherited from

		C1.LiveLinq.LiveViews.View)
	Value	Gets the value of the single element of the aggregation view.

Top

Public Methods

	Name	Description
	AsCollectionViewFactory	Returns an instance of System.ComponentModel.ICollectionViewFactory that can be used as a source of a System.Windows.Data.CollectionViewSource. (Inherited from C1.LiveLinq.LiveViews.View)
	AsDynamic	Used for views with anonymous type constructor as the result selector, converts the View to a View<dynamic> so it can be used for data binding and programmatic access. (Inherited from C1.LiveLinq.LiveViews.View)
	AttachAggregationView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)
	AttachView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)
	Concat	Concatenation of two views. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)
	Contains	Determines whether the view contains a specified item. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)
	DeferMaintenance	Enters a defer cycle that you can use to make bulk changes to the view sources and delay automatic view maintenance.

		(Inherited from C1.LiveLinq.LiveViews.View)
	GetEnumerator	Returns an enumerator that iterates through the view items. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)
	GroupBy	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)
	GroupJoin<TInner,TKey,TResult>	Correlates the elements of two views based on equality of keys and groups the results. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)
	IndexOf	Searches for the specified object among the elements of the view and returns the zero-based ordinal position of its first occurrence. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)
	Join<TInner,TKey,TResult>	Correlates the elements of two views based on matching keys. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)
	Maintain	Brings the view up to date with its source data. (Inherited from C1.LiveLinq.LiveViews.View)
	OrderBy<TKey>	Sorts the elements of a view in ascending order. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)
	OrderByDescending<TKey>	Sorts the elements of a view in descending order. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)
	PurgeEmptyGroups	Remove empty groups from a grouping view. (Inherited from C1.LiveLinq.LiveViews.View)
	Rebuild	Re-populates the view by re-executing the view's query. (Inherited from C1.LiveLinq.LiveViews.View)

	Select<TResult>	Projects each element of a view into a new form. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)
	SelectMany	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)
	SetTransaction	Sets the value of the View.Transaction property. (Inherited from C1.LiveLinq.LiveViews.View)
	ToString	Returns a string representing this view. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)
	Union	Set union of two views. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)
	Where	Filters the source view based on a predicate. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)

Top

Public Events

	Name	Description
	Changed	Occurs after an item of the view or the entire view has changed. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)

Top

See Also

Reference

AggregationView<TSource,TResult> Class
C1.LiveLinq.LiveViews Namespace

Properties
For a list of all members of this type, see AggregationView<TSource,TResult> members.

Public Properties

	Name	Description
	Count	Gets the total number of elements in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	CurrentItem	Gets the current item in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	DataBindingMode	Gets or sets the data binding mode for this view. (Inherited from C1.LiveLinq.LiveViews.View)
	DeferredMaintenance	Gets the effective value of MaintenanceMode. (Inherited from C1.LiveLinq.LiveViews.View)
	Indexes	Gets the collection of indexes for this view. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)
	IsReadOnly	Gets a value indicating whether this view is read-only, not updatable. (Inherited from C1.LiveLinq.LiveViews.View)
	Item	Gets the view item (element) at the specified ordinal position. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)
	MaintenanceMode	Gets or sets a value controlling how the view is synchronized with changes in its base data. (Inherited from C1.LiveLinq.LiveViews.View)
	MoveToFirstOnReset	Gets or sets a value indicating that the first item must be made current after initial loading or reset (on any C1.LiveLinq.SourceChangeType.Reset notification) if current item was not set by other means. The default is True. (Inherited from C1.LiveLinq.LiveViews.View)
	Order	Gets a value indicating whether and how this view preserves item order

		if it exists in its base data source. (Inherited from C1.LiveLinq.LiveViews.View)
	Transaction	Gets an instance of C1.LiveLinq.ITransaction associated with the view. If a view has a transaction associated with it, that transaction's scope is opened automatically every time the view is updated, so the programmer does not need to do it manually in code. (Inherited from C1.LiveLinq.LiveViews.View)
	Value	Gets the value of the single element of the aggregation view.

Top

See Also

Reference

AggregationView<TSource,TResult> Class
C1.LiveLinq.LiveViews Namespace

Value Property
Gets the value of the single element of the aggregation view.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Property Value As TResult
```

C#

```
public TResult Value {get;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

AggregationView<TSource,TResult> Class
AggregationView<TSource,TResult> Members

GroupView<TKey,TElement>
The type of the key used for grouping.

The type of the elements in the group view.

A group in a grouping view.

Object Model

Syntax

Visual Basic (Declaration)

`Public NotInheritable Class GroupView`
`  (Of TKey,TElement)`
`  Inherits View(Of TElement)`
`  Implements C1.LiveLinq.Indexing.IIndexedSource(Of TElement),`
`C1.LiveLinq.IObservableSource(Of TElement)`

C#

`public sealed class GroupView<TKey,TElement> : View<TElement>,`
`C1.LiveLinq.Indexing.IIndexedSource<TElement>,`
`C1.LiveLinq.IObservableSource<TElement>`

Type Parameters

TKey

The type of the key used for grouping.

TElement

The type of the elements in the group view.

Remarks

A grouping view is a result of a **GroupBy** operation on a live view. It is a collection of groups. Each group contains elements with the same key. That collection is a live view, and every group in itself is a live view, an object of type **GroupView<TKey, TElement>**.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.View
    C1.LiveLinq.LiveViews.View<T>
      C1.LiveLinq.LiveViews.GroupView<TKey,TElement>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

GroupView<TKey,TElement> Members
C1.LiveLinq.LiveViews Namespace

Overview
The type of the key used for grouping.

The type of the elements in the group view.

A group in a grouping view.

Object Model

Syntax

Visual Basic (Declaration)	

```
Public NotInheritable Class GroupView
    (Of TKey,TElement)
   Inherits View(Of TElement)
   Implements C1.LiveLinq.Indexing.IIndexedSource(Of TElement),
```

C1.LiveLinq.IObservableSource(Of TElement)
C#
`public sealed class GroupView<TKey,TElement> : View<TElement>, C1.LiveLinq.Indexing.IIndexedSource<TElement>, C1.LiveLinq.IObservableSource<TElement>`

Type Parameters

TKey

The type of the key used for grouping.

TElement

The type of the elements in the group view.

Remarks

A grouping view is a result of a **GroupBy** operation on a live view. It is a collection of groups. Each group contains elements with the same key. That collection is a live view, and every group in itself is a live view, an object of type **GroupView<TKey, TElement>**.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.View
    C1.LiveLinq.LiveViews.View<T>
      C1.LiveLinq.LiveViews.GroupView<TKey,TElement>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

GroupView<TKey,TElement> Members
C1.LiveLinq.LiveViews Namespace

Members

Properties Methods Events

The following tables list the members exposed by GroupView<TKey,TElement>.

Public Properties

	Name	Description
	Count	Gets the total number of elements in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	CurrentItem	Gets the current item in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	DataBindingMode	Gets or sets the data binding mode for this view. (Inherited from C1.LiveLinq.LiveViews.View)
	DeferredMaintenance	Overridden. This property overrides View.DeferredMaintenance.
	Indexes	Gets the collection of indexes for this view. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
	IsReadOnly	Gets a value indicating whether this view is read-only, not updatable. (Inherited from C1.LiveLinq.LiveViews.View)
	Item	Gets the view item (element) at the specified ordinal position. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
	Key	Gets the key value of the group.
	MaintenanceMode	Overridden. This property overrides View.MaintenanceMode.
	MoveToFirstOnReset	Gets or sets a value indicating that the first item must be made current after initial loading or reset (on any C1.LiveLinq.SourceChangeType.Reset notification) if current item was not set by other means. The default is True. (Inherited from

		C1.LiveLinq.LiveViews.View)
	Order	Gets a value indicating whether and how this view preserves item order if it exists in its base data source. (Inherited from C1.LiveLinq.LiveViews.View)
	Parent	Gets the grouping view (the result of a **GroupBy** operation) to which this group belongs.
	Transaction	Gets an instance of C1.LiveLinq.ITransaction associated with the view. If a view has a transaction associated with it, that transaction's scope is opened automatically every time the view is updated, so the programmer does not need to do it manually in code. (Inherited from C1.LiveLinq.LiveViews.View)

Top

Public Methods

	Name	Description
	AsCollectionViewFactory	Returns an instance of System.ComponentModel.ICollectionViewFactory that can be used as a source of a System.Windows.Data.CollectionViewSource. (Inherited from C1.LiveLinq.LiveViews.View)
	AsDynamic	Used for views with anonymous type constructor as the result selector, converts the View to a View<dynamic> so it can be used for data binding and programmatic access. (Inherited from C1.LiveLinq.LiveViews.View)
	AttachAggregationView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)

	AttachView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
	Concat	Concatenation of two views. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
	Contains	Determines whether the view contains a specified item. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
	DeferMaintenance	Enters a defer cycle that you can use to make bulk changes to the view sources and delay automatic view maintenance. (Inherited from C1.LiveLinq.LiveViews.View)
	GetEnumerator	Returns an enumerator that iterates through the view items. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
	GroupBy	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
	GroupJoin<TInner,TKey,TResult>	Correlates the elements of two views based on equality of keys and groups the results. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
	IndexOf	Searches for the specified object among the elements of the view and returns the zero-based ordinal position of its first occurrence. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
	Join<TInner,TKey,TResult>	Correlates the elements of two views based on matching keys. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
	Maintain	Overridden. This method overrides View.Maintain.
	OrderBy<TKey>	Sorts the elements of a view in ascending order. (Inherited

		from C1.LiveLinq.LiveViews.View<TElement>)
	OrderByDescending<TKey>	Sorts the elements of a view in descending order. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
	PurgeEmptyGroups	Overridden. This method overrides View.PurgeEmptyGroups.
	Rebuild	Overridden. This method overrides View.Rebuild.
	Select<TResult>	Projects each element of a view into a new form. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
	SelectMany	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
	SetTransaction	Sets the value of the View.Transaction property. (Inherited from C1.LiveLinq.LiveViews.View)
	ToString	Returns a string that represents this instance of GroupView<TKey,TElement>
	Union	Set union of two views. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
	Where	Filters the source view based on a predicate. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)

Top

Public Events

	Name	Description
	Changed	Occurs after an item of the view or the entire view has changed. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)

Top

See Also

Reference

GroupView<TKey,TElement> Class
C1.LiveLinq.LiveViews Namespace

Methods
>

Name	Description
AsCollectionViewFactory	Returns an instance of System.ComponentModel.ICollectionViewFactory that can be used as a source of a System.Windows.Data.CollectionViewSource. (Inherited from C1.LiveLinq.LiveViews.View)
AsDynamic	Used for views with anonymous type constructor as the result selector, converts the View to a View<dynamic> so it can be used for data binding and programmatic access. (Inherited from C1.LiveLinq.LiveViews.View)
AttachAggregationView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
AttachView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
Concat	Concatenation of two views. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
Contains	Determines whether the view contains a specified item. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
DeferMaintenance	Enters a defer cycle that you can use to make bulk changes to the view sources and delay automatic view maintenance. (Inherited from C1.LiveLinq.LiveViews.View)
GetEnumerator	Returns an enumerator that iterates through the view items. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)

GroupBy	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
GroupJoin<TInner,TKey,TResult>	Correlates the elements of two views based on equality of keys and groups the results. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
IndexOf	Searches for the specified object among the elements of the view and returns the zero-based ordinal position of its first occurrence. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
Join<TInner,TKey,TResult>	Correlates the elements of two views based on matching keys. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
Maintain	Overridden. This method overrides View.Maintain.
OrderBy<TKey>	Sorts the elements of a view in ascending order. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
OrderByDescending<TKey>	Sorts the elements of a view in descending order. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
PurgeEmptyGroups	Overridden. This method overrides View.PurgeEmptyGroups.
Rebuild	Overridden. This method overrides View.Rebuild.
Select<TResult>	Projects each element of a view into a new form. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
SelectMany	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
SetTransaction	Sets the value of the View.Transaction property. (Inherited from C1.LiveLinq.LiveViews.View)
ToString	Returns a string that represents this instance of GroupView<TKey,TElement>
Union	Set union of two views. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
Where	Filters the source view based on a predicate. (Inherited from

C1.LiveLinq.LiveViews.View<TElement>)

Top

See Also

Reference

GroupView<TKey,TElement> Class
C1.LiveLinq.LiveViews Namespace

Maintain Method
This method overrides View.Maintain.

Syntax

Visual Basic (Declaration)
`Public Overrides NotOverridable Sub Maintain()`
C#
`public override void Maintain()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

GroupView<TKey,TElement> Class
GroupView<TKey,TElement> Members

PurgeEmptyGroups Method
This method overrides View.PurgeEmptyGroups.

Syntax

Visual Basic (Declaration)

```
Public Overrides NotOverridable Sub PurgeEmptyGroups()
```

C#

```
public override void PurgeEmptyGroups()
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

GroupView<TKey,TElement> Class
GroupView<TKey,TElement> Members

Rebuild Method
This method overrides View.Rebuild.

Syntax

Visual Basic (Declaration)

```
Public Overrides NotOverridable Sub Rebuild()
```

C#

```
public override void Rebuild()
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

GroupView<TKey,TElement> Class
GroupView<TKey,TElement> Members

ToString Method

Returns a string that represents this instance of GroupView<TKey,TElement>

Syntax

Visual Basic (Declaration)
`Public Overrides Function ToString() As String`
C#
`public override string ToString()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

GroupView<TKey,TElement> Class
GroupView<TKey,TElement> Members

Properties

For a list of all members of this type, see GroupView<TKey,TElement> members.

Public Properties

	Name	Description
	Count	Gets the total number of elements in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	CurrentItem	Gets the current item in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	DataBindingMode	Gets or sets the data binding mode for this view. (Inherited from C1.LiveLinq.LiveViews.View)

	DeferredMaintenance	Overridden. This property overrides View.DeferredMaintenance.
	Indexes	Gets the collection of indexes for this view. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
	IsReadOnly	Gets a value indicating whether this view is read-only, not updatable. (Inherited from C1.LiveLinq.LiveViews.View)
	Item	Gets the view item (element) at the specified ordinal position. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
	Key	Gets the key value of the group.
	MaintenanceMode	Overridden. This property overrides View.MaintenanceMode.
	MoveToFirstOnReset	Gets or sets a value indicating that the first item must be made current after initial loading or reset (on any C1.LiveLinq.SourceChangeType.Reset notification) if current item was not set by other means. The default is True. (Inherited from C1.LiveLinq.LiveViews.View)
	Order	Gets a value indicating whether and how this view preserves item order if it exists in its base data source. (Inherited from C1.LiveLinq.LiveViews.View)
	Parent	Gets the grouping view (the result of a **GroupBy** operation) to which this group belongs.
	Transaction	Gets an instance of C1.LiveLinq.ITransaction associated with the view. If a view has a transaction associated with it, that transaction's scope is opened automatically every time the view is updated, so the programmer does not need to do it manually in code. (Inherited from C1.LiveLinq.LiveViews.View)

Top

See Also

Reference

GroupView<TKey,TElement> Class
C1.LiveLinq.LiveViews Namespace

DeferredMaintenance Property
This property overrides View.DeferredMaintenance.

Syntax

Visual Basic (Declaration)
`Public Overrides NotOverridable ReadOnly Property DeferredMaintenance As Boolean`
C#
`public override bool DeferredMaintenance {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

GroupView<TKey,TElement> Class
GroupView<TKey,TElement> Members

Key Property
Gets the key value of the group.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Key As TKey`
C#

```
public TKey Key {get;}
```

Remarks

The key value is common to the elements of this group.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

GroupView<TKey,TElement> Class
GroupView<TKey,TElement> Members

MaintenanceMode Property
This property overrides View.MaintenanceMode.

Syntax

Visual Basic (Declaration)
`Public Overrides NotOverridable Property MaintenanceMode As ViewMaintenanceMode`
C#
`public override ViewMaintenanceMode MaintenanceMode {get; set;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

GroupView<TKey,TElement> Class
GroupView<TKey,TElement> Members

Parent Property

Gets the grouping view (the result of a **GroupBy** operation) to which this group belongs.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Parent As View`
C#
`public View Parent {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

GroupView<TKey,TElement> Class
GroupView<TKey,TElement> Members

OrderedView<T>

The type of the elements in the view.

Represents a sorted view.

Object Model

Syntax

Visual Basic (Declaration)

`Public Class OrderedView(Of T)`
`  Inherits View(Of T)`
`  Implements C1.LiveLinq.Indexing.IIndexedSource(Of T),`
`C1.LiveLinq.IObservableSource(Of T)`

C#

```
public class OrderedView<T> : View<T>, C1.LiveLinq.Indexing.IIndexedSource<T>,
C1.LiveLinq.IObservableSource<T>
```

Type Parameters

T

The type of the elements in the view.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.View
    C1.LiveLinq.LiveViews.View<T>
      C1.LiveLinq.LiveViews.OrderedView<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

OrderedView<T> Members
C1.LiveLinq.LiveViews Namespace
OrderBy<TKey>(Expression<Func<T,TKey>>) Method
OrderByDescending<TKey> Method

Overview
The type of the elements in the view.

Represents a sorted view.

Object Model

Syntax

Visual Basic (Declaration)	

`Public Class OrderedView(Of T)`
`  Inherits View(Of T)`
`  Implements C1.LiveLinq.Indexing.IIndexedSource(Of T),`
`C1.LiveLinq.IObservableSource(Of T)`	

C#	
`public class OrderedView<T> : View<T>, C1.LiveLinq.Indexing.IIndexedSource<T>,` `C1.LiveLinq.IObservableSource<T>`	

Type Parameters

T

The type of the elements in the view.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.View
    C1.LiveLinq.LiveViews.View<T>
      C1.LiveLinq.LiveViews.OrderedView<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

OrderedView<T> Members
C1.LiveLinq.LiveViews Namespace
OrderBy<TKey>(Expression<Func<T,TKey>>) Method
OrderByDescending<TKey> Method

Members
Properties Methods Events

The following tables list the members exposed by OrderedView<T>.

Public Properties

	Name	Description
	Count	Gets the total number of elements in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	CurrentItem	Gets the current item in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	DataBindingMode	Gets or sets the data binding mode for this view. (Inherited from C1.LiveLinq.LiveViews.View)
	DeferredMaintenance	Gets the effective value of MaintenanceMode. (Inherited from C1.LiveLinq.LiveViews.View)
	Indexes	Gets the collection of indexes for this view. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	IsReadOnly	Gets a value indicating whether this view is read-only, not updatable. (Inherited from C1.LiveLinq.LiveViews.View)
	Item	Gets the view item (element) at the specified ordinal position. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	MaintenanceMode	Gets or sets a value controlling how the view is synchronized with changes in its base data. (Inherited from C1.LiveLinq.LiveViews.View)
	MoveToFirstOnReset	Gets or sets a value indicating that the first item must be made current after initial loading or reset (on any C1.LiveLinq.SourceChangeType.Reset notification) if current item was not set by other means. The default is True. (Inherited from C1.LiveLinq.LiveViews.View)
	Order	Gets a value indicating whether and how this view preserves item order

		if it exists in its base data source. (Inherited from C1.LiveLinq.LiveViews.View)
	Transaction	Gets an instance of C1.LiveLinq.ITransaction associated with the view. If a view has a transaction associated with it, that transaction's scope is opened automatically every time the view is updated, so the programmer does not need to do it manually in code. (Inherited from C1.LiveLinq.LiveViews.View)

Top

Public Methods

	Name	Description
	AsCollectionViewFactory	Returns an instance of System.ComponentModel.ICollectionViewFactory that can be used as a source of a System.Windows.Data.CollectionViewSource. (Inherited from C1.LiveLinq.LiveViews.View)
	AsDynamic	Used for views with anonymous type constructor as the result selector, converts the View to a View<dynamic> so it can be used for data binding and programmatic access. (Inherited from C1.LiveLinq.LiveViews.View)
	AttachAggregationView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	AttachView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Concat	Concatenation of two views. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Contains	Determines whether the view contains a specified item. (Inherited from C1.LiveLinq.LiveViews.View<T>)

	DeferMaintenance	Enters a defer cycle that you can use to make bulk changes to the view sources and delay automatic view maintenance. (Inherited from C1.LiveLinq.LiveViews.View)
	GetEnumerator	Returns an enumerator that iterates through the view items. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	GroupBy	Overloaded. Groups the elements of a view according to a specified key selector function. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	GroupJoin<TInner,TKey,TResult>	Correlates the elements of two views based on equality of keys and groups the results. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	IndexOf	Searches for the specified object among the elements of the view and returns the zero-based ordinal position of its first occurrence. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Join<TInner,TKey,TResult>	Correlates the elements of two views based on matching keys. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Maintain	Brings the view up to date with its source data. (Inherited from C1.LiveLinq.LiveViews.View)
	OrderBy<TKey>	Sorts the elements of a view in ascending order. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	OrderByDescending<TKey>	Sorts the elements of a view in descending order. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	PurgeEmptyGroups	Remove empty groups from a grouping view. (Inherited from C1.LiveLinq.LiveViews.View)

	Rebuild	Re-populates the view by re-executing the view's query. (Inherited from C1.LiveLinq.LiveViews.View)
	Select<TResult>	Projects each element of a view into a new form. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	SelectMany	Overloaded. Projects each element of this view to a collection of collections, flattens the resulting collections into one collection, and invokes a result selector function on each element therein. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	SetTransaction	Sets the value of the View.Transaction property. (Inherited from C1.LiveLinq.LiveViews.View)
	ThenBy<TKey>	Performs a subsequent ordering of view elements in ascending order according to a key.
	ThenByDescending<TKey>	Performs a subsequent ordering of view elements in descending order according to a key.
	ToString	Returns a string representing this view. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Union	Set union of two views. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Where	Filters the source view based on a predicate. (Inherited from C1.LiveLinq.LiveViews.View<T>)

Top

Public Events

	Name	Description

	Changed	Occurs after an item of the view or the entire view has changed. (Inherited from C1.LiveLinq.LiveViews.View<T>)

Top

See Also

Reference

OrderedView<T> Class
C1.LiveLinq.LiveViews Namespace
OrderBy<TKey>(Expression<Func<T,TKey>>) Method
OrderByDescending<TKey> Method

Methods
For a list of all members of this type, see OrderedView<T> members.

Public Methods

	Name	Description
	AsCollectionViewFactory	Returns an instance of System.ComponentModel.ICollectionViewFactory that can be used as a source of a System.Windows.Data.CollectionViewSource. (Inherited from C1.LiveLinq.LiveViews.View)
	AsDynamic	Used for views with anonymous type constructor as the result selector, converts the View to a View<dynamic> so it can be used for data binding and programmatic access. (Inherited from C1.LiveLinq.LiveViews.View)
	AttachAggregationView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	AttachView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Concat	Concatenation of two views. (Inherited from C1.LiveLinq.LiveViews.View<T>)

	Contains	Determines whether the view contains a specified item. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	DeferMaintenance	Enters a defer cycle that you can use to make bulk changes to the view sources and delay automatic view maintenance. (Inherited from C1.LiveLinq.LiveViews.View)
	GetEnumerator	Returns an enumerator that iterates through the view items. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	GroupBy	Overloaded. Groups the elements of a view according to a specified key selector function. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	GroupJoin<TInner,TKey,TResult>	Correlates the elements of two views based on equality of keys and groups the results. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	IndexOf	Searches for the specified object among the elements of the view and returns the zero-based ordinal position of its first occurrence. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Join<TInner,TKey,TResult>	Correlates the elements of two views based on matching keys. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Maintain	Brings the view up to date with its source data. (Inherited from C1.LiveLinq.LiveViews.View)
	OrderBy<TKey>	Sorts the elements of a view in ascending order. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	OrderByDescending<TKey>	Sorts the elements of a view in descending order. (Inherited from C1.LiveLinq.LiveViews.View<T>)

	PurgeEmptyGroups	Remove empty groups from a grouping view. (Inherited from C1.LiveLinq.LiveViews.View)
	Rebuild	Re-populates the view by re-executing the view's query. (Inherited from C1.LiveLinq.LiveViews.View)
	Select<TResult>	Projects each element of a view into a new form. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	SelectMany	Overloaded. Projects each element of this view to a collection of collections, flattens the resulting collections into one collection, and invokes a result selector function on each element therein. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	SetTransaction	Sets the value of the View.Transaction property. (Inherited from C1.LiveLinq.LiveViews.View)
	ThenBy<TKey>	Performs a subsequent ordering of view elements in ascending order according to a key.
	ThenByDescending<TKey>	Performs a subsequent ordering of view elements in descending order according to a key.
	ToString	Returns a string representing this view. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Union	Set union of two views. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Where	Filters the source view based on a predicate. (Inherited from C1.LiveLinq.LiveViews.View<T>)

Top

See Also

Reference

OrderedView<T> Class
C1.LiveLinq.LiveViews Namespace
OrderBy<TKey>(Expression<Func<T,TKey>>) Method
OrderByDescending<TKey> Method

ThenBy<TKey> Method
The type of the key returned by *keySelector*.

A function to extract a key from each element.

Performs a subsequent ordering of view elements in ascending order according to a key.

Syntax

Visual Basic (Declaration)

`Public Function ThenBy(Of TKey)( _`
`  ByVal keySelector As Expression(Of Func(Of T,TKey)) _`
`) As OrderedView(Of T)`

C#

`public OrderedView<T> ThenBy<TKey>(`
`  Expression<Func<T,TKey>> keySelector`
`)`

Parameters

keySelector

A function to extract a key from each element.

Type Parameters

TKey

The type of the key returned by *keySelector*.

Return Value

A view whose elements are sorted according to a key.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

OrderedView<T> Class
OrderedView<T> Members

ThenByDescending<TKey> Method
The type of the key returned by *keySelector*.

A function to extract a key from each element.

Performs a subsequent ordering of view elements in descending order according to a key.

Syntax

Visual Basic (Declaration)

`Public Function ThenByDescending(Of TKey)( _`
`  ByVal keySelector As Expression(Of Func(Of T,TKey)) _`
`) As OrderedView(Of T)`

C#

`public OrderedView<T> ThenByDescending<TKey>(`
`  Expression<Func<T,TKey>> keySelector`
`)`

Parameters

keySelector

A function to extract a key from each element.

Type Parameters

TKey

The type of the key returned by *keySelector*.

Return Value

A view whose elements are sorted in descending order according to a key.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

OrderedView<T> Class
OrderedView<T> Members

PropertyIsNotVirtualException
Represents an exception that indicates that a property used in a result selector of a live view is not virtual.

Object Model

Syntax

Visual Basic (Declaration)
`Public Class PropertyIsNotVirtualException` `  Inherits System.Exception`
C#
`public class PropertyIsNotVirtualException : System.Exception`

Inheritance Hierarchy

System.Object
  System.Exception
    C1.LiveLinq.LiveViews.PropertyIsNotVirtualException

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

PropertyIsNotVirtualException Members
C1.LiveLinq.LiveViews Namespace

Overview
Represents an exception that indicates that a property used in a result selector of a live view is not virtual.

Object Model

Syntax

Visual Basic (Declaration)
`Public Class PropertyIsNotVirtualException` `  Inherits System.Exception`
C#
`public class PropertyIsNotVirtualException : System.Exception`

Inheritance Hierarchy

System.Object
  System.Exception
    C1.LiveLinq.LiveViews.PropertyIsNotVirtualException

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

PropertyIsNotVirtualException Members
C1.LiveLinq.LiveViews Namespace

Members
Properties Methods

The following tables list the members exposed by PropertyIsNotVirtualException.

Public Properties

	Name	Description
	Context	Lambda expression where a type with the non-virtual property is used.
	Data	(Inherited from System.Exception)
	HelpLink	(Inherited from System.Exception)
	InnerException	(Inherited from System.Exception)
	Message	Overridden. Gets a message that describes the current exception.
	Property	The non-virtual property.
	ResultType	The result type of the lambda expression.
	Source	(Inherited from System.Exception)
	StackTrace	(Inherited from System.Exception)
	TargetSite	(Inherited from System.Exception)

Top

Public Methods

	Name	Description

	GetBaseException	(Inherited from System.Exception)
	GetObjectData	(Inherited from System.Exception)
	GetType	(Inherited from System.Exception)
	ToString	(Inherited from System.Exception)

Top

See Also

Reference

PropertyIsNotVirtualException Class
C1.LiveLinq.LiveViews Namespace

Properties
>

Name	Description
Context	Lambda expression where a type with the non-virtual property is used.
Data	(Inherited from System.Exception)
HelpLink	(Inherited from System.Exception)
InnerException	(Inherited from System.Exception)
Message	Overridden. Gets a message that describes the current exception.
Property	The non-virtual property.
ResultType	The result type of the lambda expression.
Source	(Inherited from System.Exception)
StackTrace	(Inherited from System.Exception)
TargetSite	(Inherited from System.Exception)

Top

See Also

Reference

PropertyIsNotVirtualException Class
C1.LiveLinq.LiveViews Namespace

Context Property
Lambda expression where a type with the non-virtual property is used.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Context As LambdaExpression`
C#
`public LambdaExpression Context {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

PropertyIsNotVirtualException Class
PropertyIsNotVirtualException Members

Message Property
Gets a message that describes the current exception.

Syntax

Visual Basic (Declaration)
`Public Overrides ReadOnly Property Message As String`
C#

```
public override string Message {get;}
```

Property Value

The error message that explains the reason for the exception.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

PropertyIsNotVirtualException Class
PropertyIsNotVirtualException Members

Property Property
The non-virtual property.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Property As PropertyInfo`
C#
`public PropertyInfo Property {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

PropertyIsNotVirtualException Class
PropertyIsNotVirtualException Members

ResultType Property

The result type of the lambda expression.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property ResultType As Type`
C#
`public Type ResultType {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

PropertyIsNotVirtualException Class
PropertyIsNotVirtualException Members

View

Base class for the View<T> class. Contains members that don't depend on the element type *T*.

Object Model

Syntax

Visual Basic (Declaration)
`Public MustInherit Class View`
C#
`public abstract class View`

Remarks

Use this class to type variables that can accept views with different element types or a view with anonymous element type.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.View
    C1.LiveLinq.LiveViews.View<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

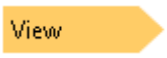
See Also

Reference

View Members
C1.LiveLinq.LiveViews Namespace

Overview
Base class for the View<T> class. Contains members that don't depend on the element type *T*.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit Class View`
C#
`public abstract class View`

Remarks

Use this class to type variables that can accept views with different element types or a view with anonymous element type.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.View
    C1.LiveLinq.LiveViews.View<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View Members
C1.LiveLinq.LiveViews Namespace

Members
Properties Methods

The following tables list the members exposed by View.

Public Properties

	Name	Description
	Count	Gets the total number of elements in the view.
	CurrentItem	Gets the current item in the view.
	DataBindingMode	Gets or sets the data binding mode for this view.
	DeferredMaintenance	Gets the effective value of MaintenanceMode.
	IsReadOnly	Gets a value indicating whether this view is read-only, not updatable.
	MaintenanceMode	Gets or sets a value controlling how the view is synchronized with changes in its base data.

	MoveToFirstOnReset	Gets or sets a value indicating that the first item must be made current after initial loading or reset (on any C1.LiveLinq.SourceChangeType.Reset notification) if current item was not set by other means. The default is True.
	Order	Gets a value indicating whether and how this view preserves item order if it exists in its base data source.
	Rows	Gets the collection of ViewRow objects used for programmatic access to view elements (items) and for data binding.
	Transaction	Gets an instance of C1.LiveLinq.ITransaction associated with the view. If a view has a transaction associated with it, that transaction's scope is opened automatically every time the view is updated, so the programmer does not need to do it manually in code.

Top

Public Methods

	Name	Description
S	AllowInResult	Specifies that a type with non-virtual properties can be used in a result selector.
	AsCollectionViewFactory	Returns an instance of System.ComponentModel.ICollectionViewFactory that can be used as a source of a System.Windows.Data.CollectionViewSource.
	AsDynamic	Used for views with anonymous type constructor as the result selector, converts the View to a View<dynamic> so it can be used for data binding and programmatic access.
	DeferMaintenance	Enters a defer cycle that you can use to make bulk changes to the view sources and delay automatic view maintenance.

	Maintain	Brings the view up to date with its source data.
	PurgeEmptyGroups	Remove empty groups from a grouping view.
	Rebuild	Re-populates the view by re-executing the view's query.
	SetTransaction	Sets the value of the Transaction property.

Top

See Also

Reference

View Class
C1.LiveLinq.LiveViews Namespace

Methods
For a list of all members of this type, see View members.

Public Methods

	Name	Description
S	AllowInResult	Specifies that a type with non-virtual properties can be used in a result selector.
	AsCollectionViewFactory	Returns an instance of System.ComponentModel.ICollectionViewFactory that can be used as a source of a System.Windows.Data.CollectionViewSource.
	AsDynamic	Used for views with anonymous type constructor as the result selector, converts the View to a View<dynamic> so it can be used for data binding and programmatic access.
	DeferMaintenance	Enters a defer cycle that you can use to make bulk changes to the view sources and delay automatic view maintenance.

	Maintain	Brings the view up to date with its source data.
	PurgeEmptyGroups	Remove empty groups from a grouping view.
	Rebuild	Re-populates the view by re-executing the view's query.
	SetTransaction	Sets the value of the Transaction property.

Top

See Also

Reference

View Class
C1.LiveLinq.LiveViews Namespace

AllowInResult Method
The type that is allowed to be used.

Specifies that a type with non-virtual properties can be used in a result selector.

Syntax

Visual Basic (Declaration)

```
Public Shared Sub AllowInResult( _
   ByVal type As Type _
)
```

C#

```
public static void AllowInResult(
   Type type
)
```

Parameters

type

The type that is allowed to be used.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View Class
View Members

AsCollectionViewFactory Method
Returns an instance of System.ComponentModel.ICollectionViewFactory that can be used as a source of a System.Windows.Data.CollectionViewSource.

Syntax

Visual Basic (Declaration)
`Public Function AsCollectionViewFactory() As ICollectionViewFactory`
C#
`public ICollectionViewFactory AsCollectionViewFactory()`

Return Value

A factory that returns the same View as a System.ComponentModel.ICollectionView.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View Class
View Members

AsDynamic Method
Used for views with anonymous type constructor as the result selector, converts the View to a View<dynamic> so it can be used for data binding and programmatic access.

Syntax

Visual Basic (Declaration)
`Public Function AsDynamic() As View(Of Object)`
C#
`public View<object> AsDynamic()`

Return Value

A dynamic view.

Remarks

A view with anonymous type constructor as the result selector cannot be used for data binding or programmatic access without applying AsDynamic to it. An attempt to do so results in an exception. After applying AsDynamic, such view can be used for data binding and programmatic access without limitations.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View Class
View Members

DeferMaintenance Method
Enters a defer cycle that you can use to make bulk changes to the view sources and delay automatic view maintenance.

Syntax

Visual Basic (Declaration)
`Public Function DeferMaintenance() As IDisposable`

C#
`public IDisposable DeferMaintenance()`

Return Value

An System.IDisposable object that you can use to dispose of the calling object.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View Class
View Members

Maintain Method
Brings the view up to date with its source data.

Syntax

Visual Basic (Declaration)
`Public Overridable Sub Maintain()`
C#
`public virtual void Maintain()`

Remarks

If source data have not changed since the last time the view was maintained (updated), this method does nothing. It also does nothing if the view's MaintenanceMode is **Immediate**, because in that case the view is guaranteed to be in synch with its base data at all times. If the view is in deferred mode (its MaintenanceMode property returns **true**), the programmer can use the **Maintain** method to force updating the view.

Note that it is not necessary to call **Maintain** to make sure you get updated data from the view. The view is automatically updated every time you request data from it, if base data changed since the last request, regardless of the view's MaintenanceMode.

LiveLinq maintains views using optimized incremental algorithms, not simply re-populates them from scratch. It calculating the delta in the view from the delta in the base data. In most cases, it allows to propagate the change from base data to the view very fast.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View Class
View Members

PurgeEmptyGroups Method
Remove empty groups from a grouping view.

Syntax

Visual Basic (Declaration)
`Public Overridable Sub PurgeEmptyGroups()`
C#
`public virtual void PurgeEmptyGroups()`

Remarks

This method is used only for **GroupBy** (grouping) views, does nothing for views of other kinds.

When a grouping view is populated, it does not contain empty groups. But later, as a result of maintaining the grouping view, keeping it in synch with changes in its base data, some of the groups can become empty. If it is undesirable to have empty groups, you can call this method to delete them.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View Class
View Members

Rebuild Method
Re-populates the view by re-executing the view's query.

Syntax

Visual Basic (Declaration)
`Public Overridable Sub Rebuild()`
C#
`public virtual void Rebuild()`

Remarks

This method is rarely needed, because normally automatic incremental maintenance is faster than re-executing the query over the entire base data collection. However, if for some reason you need to re-populate it from scratch, that can be done with the **Rebuild** method.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View Class
View Members

SetTransaction Method
The new value for the the Transaction property.

Set this parameter to True to prevent exception if you have writable property paths and want to ignore them (but be aware that updates through property paths are not tracked by the transaction). The default is False.

Sets the value of the Transaction property.

Syntax

Visual Basic (Declaration)

```
Public Sub SetTransaction( _
   ByVal transaction As ITransaction, _
   Optional ByVal allowPropertyPaths As Boolean _
)
```

C#

```
public void SetTransaction(
   ITransaction transaction,
   bool allowPropertyPaths
)
```

Parameters

transaction

The new value for the the Transaction property.

allowPropertyPaths

Set this parameter to True to prevent exception if you have writable property paths and want to ignore them (but be aware that updates through property paths are not tracked by the transaction). The default is False.

Remarks

If a writable property path exists in the view element type (for example, `Order.Customer.City`), then an exception is thrown unless suppressed by setting allowPropertyPaths to True, because modifying properties using such paths cannot be supported by transactions. To prevent the exception, set the allowPropertyPaths parameter to True, but make sure you do not use such property paths in your code and don't have a two-way data binding to such property path in your controls. If you modify a property using such path in your code or through data binding, that modification happens outside the transaction scope.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View Class
View Members

Properties
For a list of all members of this type, see View members.

Public Properties

	Name	Description
	Count	Gets the total number of elements in the view.
	CurrentItem	Gets the current item in the view.
	DataBindingMode	Gets or sets the data binding mode for this view.
	DeferredMaintenance	Gets the effective value of MaintenanceMode.
	IsReadOnly	Gets a value indicating whether this view is read-only, not updatable.
	MaintenanceMode	Gets or sets a value controlling how the view is synchronized with changes in its base data.
	MoveToFirstOnReset	Gets or sets a value indicating that the first item must be made current after initial loading or reset (on any C1.LiveLinq.SourceChangeType.Reset notification) if current item was not set by other means. The default is True.
	Order	Gets a value indicating whether and how this view preserves item order if it exists in its base data source.

	Rows	Gets the collection of ViewRow objects used for programmatic access to view elements (items) and for data binding.
	Transaction	Gets an instance of C1.LiveLinq.ITransaction associated with the view. If a view has a transaction associated with it, that transaction's scope is opened automatically every time the view is updated, so the programmer does not need to do it manually in code.

Top

See Also

Reference

View Class
C1.LiveLinq.LiveViews Namespace

Count Property
Gets the total number of elements in the view.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Property Count As Integer
```

C#

```
public int Count {get;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View Class
View Members

CurrentItem Property

Gets the current item in the view.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property CurrentItem As Object`
C#
`public object CurrentItem {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View Class

View Members

DataBindingMode Property

Gets or sets the data binding mode for this view.

Syntax

Visual Basic (Declaration)
`Public Property DataBindingMode As DataBindingMode`
C#
`public DataBindingMode DataBindingMode {get; set;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View Class
View Members
DataBindingMode enumeration.

DeferredMaintenance Property
Gets the effective value of MaintenanceMode.

Syntax

Visual Basic (Declaration)
`Public Overridable ReadOnly Property DeferredMaintenance As Boolean`
C#
`public virtual bool DeferredMaintenance {get;}`

Property Value

true if MaintenanceMode is set to **Deferred**, or if it is set to **Default** and no listeners are registered with this view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View Class
View Members

IsReadOnly Property
Gets a value indicating whether this view is read-only, not updatable.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property IsReadOnly As Boolean`
C#
`public bool IsReadOnly {get;}`

Remarks

Properties exposed by a view can be updatable or read-only. Updatable properties of a view can be modified directly in the view.

All properties of a read-only (not updatable) view are read-only. In an updatable view, properties directly corresponding to base data (source) properties are updatable, calculated properties are read-only. For example, in a view `from x in X select new { x.P, Q = x.Q + 1 }` P is updatable and Q is read-only.

Read-only properties of a view cannot be modified directly in the view, but they still reflect up-to-date values of the source, so the difference is often not critical, you can always modify corresponding property in the source, that will automatically change the property in the view.

Also, a read-only view cannot be cleared or its rows deleted or new rows added directly in the view (but all these actions can be performed on the source data collection and they will normally result in the corresponding changes of the view).

A **Join** view is read-only by default, but one of its two parts can be made updatable using the C1.LiveLinq.LiveViewExtensions.AsUpdatable<T> method.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View Class
View Members
AsUpdatable<T> Method
ViewRow Class
ViewRowState Enumeration

MaintenanceMode Property

Gets or sets a value controlling how the view is synchronized with changes in its base data.

Syntax

Visual Basic (Declaration)
`Public Overridable Property MaintenanceMode As ViewMaintenanceMode`
C#
`public virtual ViewMaintenanceMode MaintenanceMode {get; set;}`

Remarks

A view in **Default** mode (which is the default value for this property) is effectively in **Immediate** mode if it has a listener (for example, if a GUI control is bound to it); otherwise it is in **Deferred** mode. To find out whether the view is effectively in **Deferred** or in **Immediate** mode, you can use the DeferredMaintenance property.

If you set this property to **Deferred**, no listeners are allowed to register with this view. An attempt to register a listener will result in an exception.

See Also: View Maintenance Mode.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View Class
View Members
DeferredMaintenance Property

MoveToFirstOnReset Property

Gets or sets a value indicating that the first item must be made current after initial loading or reset (on any C1.LiveLinq.SourceChangeType.Reset notification) if current item was not set by other means. The default is True.

Syntax

Visual Basic (Declaration)
`Public Property MoveToFirstOnReset As Boolean`
C#
`public bool MoveToFirstOnReset {get; set;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View Class
View Members

Order Property
Gets a value indicating whether and how this view preserves item order if it exists in its base data source.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Order As ViewOrder`
C#
`public ViewOrder Order {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View Class
View Members

Rows Property
Gets the collection of ViewRow objects used for programmatic access to view elements (items) and for data binding.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Rows As ViewRowCollection`
C#
`public ViewRowCollection Rows {get;}`

Remarks

There can be a view with elements of any type *T*, as represented by the generic class View<T>. That type is not always known beforehand, so it is not always possible to access properties of view elements with strong-typed (early binding) code. Dynamic (untyped, late binding) access to view elements is provided by the ViewRowCollection owned by the view.

The collection of *view rows* (ViewRow objects) is always synchronized with the collection of view elements. ViewRow objects provide programmatic access to view elements and their properties.

Also, in WinForms, the **Rows** collection serves as the data source for data binding, when you bind a control or another client to a view, because a view returns its **Rows** collection in its implementation of the System.ComponentModel.IListSource interface.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View Class
View Members

Transaction Property

Gets an instance of C1.LiveLinq.ITransaction associated with the view. If a view has a transaction associated with it, that transaction's scope is opened automatically every time the view is updated, so the programmer does not need to do it manually in code.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Transaction As ITransaction`
C#
`public ITransaction Transaction {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

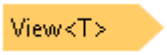
View Class
View Members

View<T>

The type of the elements in the view.

Represents a *live view*: a LINQ query result that supports two-way data binding and is kept up-to-date with base data.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class View(Of T)`

```
   Inherits View
   Implements C1.LiveLinq.Indexing.IIndexedSource(Of T),
C1.LiveLinq.IObservableSource(Of T)
```

C#

```
public class View<T> : View, C1.LiveLinq.Indexing.IIndexedSource<T>,
C1.LiveLinq.IObservableSource<T>
```

Type Parameters

T

The type of the elements in the view.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.View
    C1.LiveLinq.LiveViews.View<T>
      C1.Data.ClientView<T>
      C1.LiveLinq.LiveViews.AggregationView<TSource,TResult>
      C1.LiveLinq.LiveViews.GroupView<TKey,TElement>
      C1.LiveLinq.LiveViews.OrderedView<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

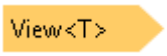
Reference

View<T> Members
C1.LiveLinq.LiveViews Namespace

Overview
The type of the elements in the view.

Represents a *live view*: a LINQ query result that supports two-way data binding and is kept up-to-date with base data.

Object Model



Syntax

Visual Basic (Declaration)

`Public Class View(Of T)`
`  Inherits View`
`  Implements C1.LiveLinq.Indexing.IIndexedSource(Of T),`
`C1.LiveLinq.IObservableSource(Of T)`

C#
`public class View<T> : View, C1.LiveLinq.Indexing.IIndexedSource<T>,` `C1.LiveLinq.IObservableSource<T>`

Type Parameters

T

The type of the elements in the view.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.View
    C1.LiveLinq.LiveViews.View<T>
      C1.Data.ClientView<T>
      C1.LiveLinq.LiveViews.AggregationView<TSource,TResult>
      C1.LiveLinq.LiveViews.GroupView<TKey,TElement>
      C1.LiveLinq.LiveViews.OrderedView<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Members
C1.LiveLinq.LiveViews Namespace

Members
Properties Methods Events

The following tables list the members exposed by View<T>.

Public Properties

	Name	Description
	Count	Gets the total number of elements in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	CurrentItem	Gets the current item in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	DataBindingMode	Gets or sets the data binding mode for this view. (Inherited from C1.LiveLinq.LiveViews.View)
	DeferredMaintenance	Gets the effective value of MaintenanceMode. (Inherited from C1.LiveLinq.LiveViews.View)
	Indexes	Gets the collection of indexes for this view.
	IsReadOnly	Gets a value indicating whether this view is read-only, not updatable. (Inherited from C1.LiveLinq.LiveViews.View)
	Item	Gets the view item (element) at the specified ordinal position.
	MaintenanceMode	Gets or sets a value controlling how the view is synchronized with changes in its base data. (Inherited from C1.LiveLinq.LiveViews.View)
	MoveToFirstOnReset	Gets or sets a value indicating that the first item must be made current after initial loading or reset (on any C1.LiveLinq.SourceChangeType.Reset notification) if current item was not set by other means. The default is True. (Inherited from

		C1.LiveLinq.LiveViews.View)
	Order	Gets a value indicating whether and how this view preserves item order if it exists in its base data source. (Inherited from C1.LiveLinq.LiveViews.View)
	Transaction	Gets an instance of C1.LiveLinq.ITransaction associated with the view. If a view has a transaction associated with it, that transaction's scope is opened automatically every time the view is updated, so the programmer does not need to do it manually in code. (Inherited from C1.LiveLinq.LiveViews.View)

Top

Public Methods

	Name	Description
	AsCollectionViewFactory	Returns an instance of System.ComponentModel.ICollectionViewFactory that can be used as a source of a System.Windows.Data.CollectionViewSource. (Inherited from C1.LiveLinq.LiveViews.View)
	AsDynamic	Used for views with anonymous type constructor as the result selector, converts the View to a View<dynamic> so it can be used for data binding and programmatic access. (Inherited from C1.LiveLinq.LiveViews.View)
	AttachAggregationView	Overloaded.
	AttachView	Overloaded.
	Concat	Concatenation of two views.

	Contains	Determines whether the view contains a specified item.
	DeferMaintenance	Enters a defer cycle that you can use to make bulk changes to the view sources and delay automatic view maintenance. (Inherited from C1.LiveLinq.LiveViews.View)
	GetEnumerator	Returns an enumerator that iterates through the view items.
	GroupBy	Overloaded. Groups the elements of a view according to a specified key selector function.
	GroupJoin<TInner,TKey,TResult>	Correlates the elements of two views based on equality of keys and groups the results.
	IndexOf	Searches for the specified object among the elements of the view and returns the zero-based ordinal position of its first occurrence.
	Join<TInner,TKey,TResult>	Correlates the elements of two views based on matching keys.
	Maintain	Brings the view up to date with its source data. (Inherited from C1.LiveLinq.LiveViews.View)
	OrderBy<TKey>	Sorts the elements of a view in ascending order.
	OrderByDescending<TKey>	Sorts the elements of a view in descending order.
	PurgeEmptyGroups	Remove empty groups from a grouping view. (Inherited from C1.LiveLinq.LiveViews.View)
	Rebuild	Re-populates the view by re-executing the view's query. (Inherited from C1.LiveLinq.LiveViews.View)
	Select<TResult>	Projects each element of a view into a new form.

	SelectMany	Overloaded. Projects each element of this view to a collection of collections, flattens the resulting collections into one collection, and invokes a result selector function on each element therein.
	SetTransaction	Sets the value of the Transaction property. (Inherited from C1.LiveLinq.LiveViews.View)
	ToString	Returns a string representing this view.
	Union	Set union of two views.
	Where	Filters the source view based on a predicate.

Top

Public Events

	Name	Description
	Changed	Occurs after an item of the view or the entire view has changed.

Top

See Also

Reference

View<T> Class
C1.LiveLinq.LiveViews Namespace

Methods
For a list of all members of this type, see View<T> members.

Public Methods

	Name	Description
	AsCollectionViewFactory	Returns an instance of

		System.ComponentModel.ICollectionViewFactory that can be used as a source of a System.Windows.Data.CollectionViewSource. (Inherited from C1.LiveLinq.LiveViews.View)
	AsDynamic	Used for views with anonymous type constructor as the result selector, converts the View to a View<dynamic> so it can be used for data binding and programmatic access. (Inherited from C1.LiveLinq.LiveViews.View)
	AttachAggregationView	Overloaded.
	AttachView	Overloaded.
	Concat	Concatenation of two views.
	Contains	Determines whether the view contains a specified item.
	DeferMaintenance	Enters a defer cycle that you can use to make bulk changes to the view sources and delay automatic view maintenance. (Inherited from C1.LiveLinq.LiveViews.View)
	GetEnumerator	Returns an enumerator that iterates through the view items.
	GroupBy	Overloaded. Groups the elements of a view according to a specified key selector function.
	GroupJoin<TInner,TKey,TResult>	Correlates the elements of two views based on equality of keys and groups the results.
	IndexOf	Searches for the specified object among the elements of the view and returns the zero-based ordinal position of its first occurrence.
	Join<TInner,TKey,TResult>	Correlates the elements of two views based on matching

		keys.
	Maintain	Brings the view up to date with its source data. (Inherited from C1.LiveLinq.LiveViews.View)
	OrderBy<TKey>	Sorts the elements of a view in ascending order.
	OrderByDescending<TKey>	Sorts the elements of a view in descending order.
	PurgeEmptyGroups	Remove empty groups from a grouping view. (Inherited from C1.LiveLinq.LiveViews.View)
	Rebuild	Re-populates the view by re-executing the view's query. (Inherited from C1.LiveLinq.LiveViews.View)
	Select<TResult>	Projects each element of a view into a new form.
	SelectMany	Overloaded. Projects each element of this view to a collection of collections, flattens the resulting collections into one collection, and invokes a result selector function on each element therein.
	SetTransaction	Sets the value of the Transaction property. (Inherited from C1.LiveLinq.LiveViews.View)
	ToString	Returns a string representing this view.
	Union	Set union of two views.
	Where	Filters the source view based on a predicate.

Top

See Also

Reference

View<T> Class
C1.LiveLinq.LiveViews Namespace

AttachAggregationView Method

Overload List

Overload	Description
AttachAggregationView<TResult>(Object,Func<View,AggregationView<T,TResult>>)	
AttachAggregationView<TResult>(Func<View,AggregationView<T,TResult>>)	

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members

AttachAggregationView<TResult>(Object,Func<View,AggregationView<T,TResult>>) Method

Syntax

Visual Basic (Declaration)

```
Public Overloads Function AttachAggregationView(Of TResult)( _
   ByVal subqueryId As Object, _
   ByVal selector As Func(Of View(Of T),AggregationView(Of T,TResult)) _
) As AggregationView(Of T,TResult)
```

C#

```
public AggregationView<T,TResult> AttachAggregationView<TResult>(
   object subqueryId,
   Func<View<T>,AggregationView<T,TResult>> selector
)
```

Parameters

subqueryId

selector

Type Parameters

TResult

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members
Overload List

AttachAggregationView<TResult>(Func<View,AggregationView<T,TResult>>) Method

Syntax

Visual Basic (Declaration)

```
Public Overloads Function AttachAggregationView(Of TResult)( _
   ByVal selector As Func(Of View(Of T),AggregationView(Of T,TResult)) _
) As AggregationView(Of T,TResult)
```

C#

```
public AggregationView<T,TResult> AttachAggregationView<TResult>(
   Func<View<T>,AggregationView<T,TResult>> selector
)
```

Parameters

selector

Type Parameters

TResult

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members
Overload List

AttachView Method

Overload List

Overload	Description
AttachView<TResult>(Func<View,View<TResult>>)	
AttachView<TResult>(Object,Func<View,View<TResult>>)	

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members

AttachView<TResult>(Func<View,View<TResult>>) Method

Syntax

Visual Basic (Declaration)
`Public Overloads Function AttachView(Of TResult)( _`

```
   ByVal selector As Func(Of View(Of T),View(Of TResult)) _
) As View(Of TResult)
```

C#

```
public View<TResult> AttachView<TResult>(
   Func<View<T>,View<TResult>> selector
)
```

Parameters

selector

Type Parameters

TResult

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members
Overload List

AttachView<TResult>(Object,Func<View,View<TResult>>) Method

Syntax

Visual Basic (Declaration)

```
Public Overloads Function AttachView(Of TResult)( _
   ByVal subqueryId As Object, _
   ByVal selector As Func(Of View(Of T),View(Of TResult)) _
) As View(Of TResult)
```

C#

```
public View<TResult> AttachView<TResult>(
```

```
   object subqueryId,
   Func<View<T>,View<TResult>> selector
)
```

Parameters

subqueryId

selector

Type Parameters

TResult

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members
Overload List

Concat Method

A collection (usually, a view) to concatenate to this view's collection of elements.

Concatenation of two views.

Syntax

Visual Basic (Declaration)

```
Public Function Concat( _
   ByVal second As IObservableSource(Of T) _
) As View(Of T)
```

C#

```
public View<T> Concat(
   IObservableSource<T> second
```

```
)
```

Parameters

second

A collection (usually, a view) to concatenate to this view's collection of elements.

Return Value

The view that contains first the elements of this view and then the elements of the parameter collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members

Contains Method
The item to locate in the view.

Determines whether the view contains a specified item.

Syntax

Visual Basic (Declaration)

```
Public Function Contains( _
   ByVal item As T _
) As Boolean
```

C#

```
public bool Contains(
   T item
)
```

Parameters

item

The item to locate in the view.

Return Value

true if the view contains the specified item; otherwise, **false**.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members

GetEnumerator Method

Returns an enumerator that iterates through the view items.

Syntax

Visual Basic (Declaration)
`Public Function GetEnumerator() As IEnumerator(Of T)`
C#
`public IEnumerator<T> GetEnumerator()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members

GroupBy Method
Groups the elements of a view according to a specified key selector function.

Overload List

Overload	Description
GroupBy<TKey>(Expression<Func<T,TKey>>)	Groups the elements of a view according to a specified key selector function.
GroupBy<TKey,TElement>(Expression<Func<T,TKey>>,Expression<Func<T,TElement>>)	Groups the elements of a view according to a specified key

	select or function and projects the elements for each group by using a specified function.
GroupBy<TKey,TElement,TResult>(Expression<Func<T,TKey>>,Expression<Func<T,TElement>>,Expression<Func<TKey,IEnumerable<TElement>,TResult>>)	Groups the elements of a view according to a specified key selector function and

	creates a result value from each group and its key. The elements of each group are projected by using a specified function.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members

The type of the key returned by *keySelector*.

A function to extract the key for each element.

Groups the elements of a view according to a specified key selector function.

Syntax

Visual Basic (Declaration)

`Public Overloads Function GroupBy(Of TKey)( _`
`  ByVal keySelector As Expression(Of Func(Of T,TKey)) _`
`) As View(Of GroupView(Of TKey,T))`

C#

`public View<GroupView<TKey,T>> GroupBy<TKey>(`
`  Expression<Func<T,TKey>> keySelector`
`)`

Parameters

keySelector

A function to extract the key for each element.

Type Parameters

TKey

The type of the key returned by *keySelector*.

Return Value

A view containing elements of type GroupView<TKey,TElement> each containing a key value and a view of the elements having that key value.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members
Overload List

GroupBy<TKey,TElement>(Expression<Func<T,TKey>>,Expression<Func<T,TElement>>) Method
The type of the key returned by *keySelector*.

The type of the element to which elements of each group are projected.

A function to extract the key for each element.

A function to map each source element to a *TElement*.

Groups the elements of a view according to a specified key selector function and projects the elements for each group by using a specified function.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function GroupBy
    (Of TKey,TElement)( _
   ByVal keySelector As Expression(Of Func(Of T,TKey)), _
   ByVal elementSelector As Expression(Of Func(Of T,TElement)) _
) As View(Of GroupView(Of TKey,TElement))
```

C#

```
public View<GroupView<TKey,TElement>> GroupBy<TKey,TElement>(
   Expression<Func<T,TKey>> keySelector,
   Expression<Func<T,TElement>> elementSelector
)
```

Parameters

keySelector

A function to extract the key for each element.

elementSelector

A function to map each source element to a *TElement*.

Type Parameters

TKey

The type of the key returned by *keySelector*.

TElement

The type of the element to which elements of each group are projected.

Return Value

A view containing elements of type GroupView<TKey,TElement> each containing a key value and a view of the elements projected (mapped) from the elements having that key value.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members
Overload List

GroupBy<TKey,TElement,TResult>(Expression<Func<T,TKey>>,Expression<Func<T,TElement>>,Expression<Func<TKey,IEnumerable<TElement>,TResult>>) Method

The type of the key returned by *keySelector*.

The type of the elements in groups.

The type of the result value returned by *resultSelector*.

A function to extract the key for each element.

A function to map each source element to an element in the IGrouping.

A function to create a result value from each group.

Groups the elements of a view according to a specified key selector function and creates a result value from each group and its key. The elements of each group are projected by using a specified function.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function GroupBy
    (Of TKey,TElement,TResult)( _
   ByVal keySelector As Expression(Of Func(Of T,TKey)), _
   ByVal elementSelector As Expression(Of Func(Of T,TElement)), _
   ByVal resultSelector As Expression(Of Func(Of TKey,IEnumerable(Of
TElement),TResult)) _
) As View(Of TResult)
```

C#

```
public View<TResult> GroupBy<TKey,TElement,TResult>(
   Expression<Func<T,TKey>> keySelector,
   Expression<Func<T,TElement>> elementSelector,
   Expression<Func<TKey,IEnumerable<TElement>,TResult>> resultSelector
)
```

Parameters

keySelector

A function to extract the key for each element.

elementSelector

A function to map each source element to an element in the IGrouping.

resultSelector

A function to create a result value from each group.

Type Parameters

TKey

The type of the key returned by *keySelector*.

TElement

The type of the elements in groups.

TResult

The type of the result value returned by *resultSelector*.

Return Value

A view of elements of type *TResult* where each element represents a projection over a group and its key.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members
Overload List

GroupJoin<TInner,TKey,TResult> Method
The type of the elements of the view to join with this view.

The type of the keys returned by the key selector functions.

The type of the result elements.

The collection (usually, a view) to join to this view.

A function to extract the join key from each element of this view.

A function to extract the join key from each element of the second view.

A function to create a result view from an element from this view and a collection of matching elements from the second view.

Correlates the elements of two views based on equality of keys and groups the results.

Syntax

Visual Basic (Declaration)

```
Public Function GroupJoin
    (Of TInner,TKey,TResult)( _
   ByVal inner As IObservableSource(Of TInner), _
   ByVal outerKeySelector As Expression(Of Func(Of T,TKey)), _
   ByVal innerKeySelector As Expression(Of Func(Of TInner,TKey)), _
```

```
   ByVal resultSelector As Expression(Of Func(Of T,GroupView(Of
TKey,TInner),TResult)) _
) As View(Of TResult)
```

C#

```
public View<TResult> GroupJoin<TInner,TKey,TResult>(
   IObservableSource<TInner> inner,
   Expression<Func<T,TKey>> outerKeySelector,
   Expression<Func<TInner,TKey>> innerKeySelector,
   Expression<Func<T,GroupView<TKey,TInner>,TResult>> resultSelector
)
```

Parameters

inner

The collection (usually, a view) to join to this view.

outerKeySelector

A function to extract the join key from each element of this view.

innerKeySelector

A function to extract the join key from each element of the second view.

resultSelector

A function to create a result view from an element from this view and a collection of matching elements from the second view.

Type Parameters

TInner

The type of the elements of the view to join with this view.

TKey

The type of the keys returned by the key selector functions.

TResult

The type of the result elements.

Return Value

A view containing elements of type *TResult* that are obtained by performing a grouped join on two views.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members

IndexOf Method
The object to locate in the view.

Searches for the specified object among the elements of the view and returns the zero-based ordinal position of its first occurrence.

Syntax

Visual Basic (Declaration)

```
Public Function IndexOf( _
   ByVal item As T _
) As Integer
```

C#

```
public int IndexOf(
   T item
)
```

Parameters

item

The object to locate in the view.

Return Value

The zero-based ordinal position of the first occurrence of *item* in the view, if found; otherwise, -1.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members

Join<TInner,TKey,TResult> Method
The type of the elements of the view to join with this view.

The type of the keys returned by the key selector functions.

The type of the result elements.

The collection (usually, a view) to join to this view.

A function to extract the join key from each element of this view.

A function to extract the join key from each element of the second view.

A function to create a result element from two matching elements.

Correlates the elements of two views based on matching keys.

Syntax

Visual Basic (Declaration)

```
Public Function Join
    (Of TInner,TKey,TResult)( _
   ByVal inner As IObservableSource(Of TInner), _
   ByVal outerKeySelector As Expression(Of Func(Of T,TKey)), _
   ByVal innerKeySelector As Expression(Of Func(Of TInner,TKey)), _
   ByVal resultSelector As Expression(Of Func(Of T,TInner,TResult)) _
) As View(Of TResult)
```

C#

```
public View<TResult> Join<TInner,TKey,TResult>(
   IObservableSource<TInner> inner,
   Expression<Func<T,TKey>> outerKeySelector,
   Expression<Func<TInner,TKey>> innerKeySelector,
   Expression<Func<T,TInner,TResult>> resultSelector
)
```

Parameters

inner

The collection (usually, a view) to join to this view.

outerKeySelector

A function to extract the join key from each element of this view.

innerKeySelector

A function to extract the join key from each element of the second view.

resultSelector

A function to create a result element from two matching elements.

Type Parameters

TInner

The type of the elements of the view to join with this view.

TKey

The type of the keys returned by the key selector functions.

TResult

The type of the result elements.

Return Value

A view containing elements of type *TResult* that are obtained by performing an inner join on two views.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members

OrderBy<TKey>(Expression<Func<T,TKey>>) Method
The type of the key returned by *keySelector*.

A function to extract a key from an element.

Sorts the elements of a view in ascending order.

Syntax

Visual Basic (Declaration)

`Public Function OrderBy(Of TKey)( _`
`  ByVal keySelector As Expression(Of Func(Of T,TKey)) _`
`) As OrderedView(Of T)`

C#

`public OrderedView<T> OrderBy<TKey>(`
`  Expression<Func<T,TKey>> keySelector`
`)`

Parameters

keySelector

A function to extract a key from an element.

Type Parameters

TKey

The type of the key returned by *keySelector*.

Return Value

A view whose elements are sorted according to a key.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members

OrderByDescending<TKey> Method
The type of the key returned by *keySelector*.

A function to extract a key from an element.

Sorts the elements of a view in descending order.

Syntax

Visual Basic (Declaration)

`Public Function OrderByDescending(Of TKey)( _`
`  ByVal keySelector As Expression(Of Func(Of T,TKey)) _`
`) As OrderedView(Of T)`

C#

`public OrderedView<T> OrderByDescending<TKey>(`
`  Expression<Func<T,TKey>> keySelector`
`)`

Parameters

keySelector

A function to extract a key from an element.

Type Parameters

TKey

The type of the key returned by *keySelector*.

Return Value

A view whose elements are sorted in descending order according to a key.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members

Select<TResult> Method
The type of the value returned by *selector*.

A transform function to apply to each element.

Projects each element of a view into a new form.

Syntax

Visual Basic (Declaration)

```
Public Function Select(Of TResult)( _
   ByVal selector As Expression(Of Func(Of T,TResult)) _
) As View(Of TResult)
```

C#

```
public View<TResult> Select<TResult>(
   Expression<Func<T,TResult>> selector
)
```

Parameters

selector

A transform function to apply to each element.

Type Parameters

TResult

The type of the value returned by *selector*.

Return Value

A view whose elements are the result of invoking the transform function on each element of this view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members

SelectMany Method
Projects each element of this view to a collection of collections, flattens the resulting collections into one collection, and invokes a result selector function on each element therein.

Overload List

Overload	Description
SelectMany<TCollection,TResult>(Expression<Func<T,IObservableSource<TCollection>>>,Expression<Func<T,TCollection,TResult>>)	Projects each element of this view to a collection of collecti

	ons, flattens the resulting collections into one collection, and invokes a result selector function on each element therein.
SelectMany<TResult>(Expression<Func<T,IObservableSource<TResult>>>)	Projects each element of this view to a collection of *TResult* and flattens

	the resulting collections into one view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members

SelectMany<TCollection,TResult>(Expression<Func<T,IObservableSource<TCollection>>>,Expression<Func<T,TCollection,TResult>>) Method
The type of the intermediate elements collected by *collectionSelector*.

The type of the elements of the resulting view.

A transform function to apply to each element of this view.

A transform function to apply to each element of the intermediate collection.

Projects each element of this view to a collection of collections, flattens the resulting collections into one collection, and invokes a result selector function on each element therein.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function SelectMany
    (Of TCollection,TResult)( _
   ByVal collectionSelector As Expression(Of Func(Of T,IObservableSource(Of
```

```
TCollection))), _
   ByVal resultSelector As Expression(Of Func(Of T,TCollection,TResult)) _
) As View(Of TResult)
```

C#

```
public View<TResult> SelectMany<TCollection,TResult>(
   Expression<Func<T,IObservableSource<TCollection>>> collectionSelector,
   Expression<Func<T,TCollection,TResult>> resultSelector
)
```

Parameters

collectionSelector

A transform function to apply to each element of this view.

resultSelector

A transform function to apply to each element of the intermediate collection.

Type Parameters

TCollection

The type of the intermediate elements collected by *collectionSelector*.

TResult

The type of the elements of the resulting view.

Return Value

A view whose elements are the result of invoking the one-to-many transform function *collectionSelector* on each element of this view and then mapping each of those collection elements and their corresponding source element to a result element.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members
Overload List

SelectMany<TResult>(Expression<Func<T,IObservableSource<TResult>>>) Method
The type of the elements of the resulting view.

A transform function to apply to each element.

Projects each element of this view to a collection of *TResult* and flattens the resulting collections into one view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function SelectMany(Of TResult)( _
   ByVal selector As Expression(Of Func(Of T,IObservableSource(Of TResult))) _
) As View(Of TResult)
```

C#

```
public View<TResult> SelectMany<TResult>(
   Expression<Func<T,IObservableSource<TResult>>> selector
)
```

Parameters

selector

A transform function to apply to each element.

Type Parameters

TResult

The type of the elements of the resulting view.

Return Value

A view whose elements are the result of invoking the one-to-many transform function on each element of this view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members
Overload List

ToString Method
Returns a string representing this view.

Syntax

Visual Basic (Declaration)
`Public Overrides Function ToString() As String`
C#
`public override string ToString()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members

Union Method
A collection (usually, a view) whose distinct elements form the second set for the union.

Set union of two views.

Syntax

Visual Basic (Declaration)

```
Public Function Union( _
   ByVal second As IObservableSource(Of T) _
) As View(Of T)
```

C#

```
public View<T> Union(
   IObservableSource<T> second
)
```

Parameters

second

A collection (usually, a view) whose distinct elements form the second set for the union.

Return Value

The view that contains the elements from both input views, excluding duplicates.

Remarks

This method excludes duplicates from the result set. This is different behavior to the Concat method, which returns all the elements in the input sequences including duplicates.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members

Where Method
A function to test each element for a condition.

Filters the source view based on a predicate.

Syntax

Visual Basic (Declaration)

```
Public Function Where( _
   ByVal predicate As Expression(Of Func(Of T,Boolean)) _
) As View(Of T)
```

C#

```
public View<T> Where(
   Expression<Func<T,bool>> predicate
)
```

Parameters

predicate

A function to test each element for a condition.

Return Value

A view that contains elements of this view that satisfy the condition.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members

Properties
For a list of all members of this type, see View<T> members.

Public Properties

	Name	Description

	Count	Gets the total number of elements in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	CurrentItem	Gets the current item in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	DataBindingMode	Gets or sets the data binding mode for this view. (Inherited from C1.LiveLinq.LiveViews.View)
	DeferredMaintenance	Gets the effective value of MaintenanceMode. (Inherited from C1.LiveLinq.LiveViews.View)
	Indexes	Gets the collection of indexes for this view.
	IsReadOnly	Gets a value indicating whether this view is read-only, not updatable. (Inherited from C1.LiveLinq.LiveViews.View)
	Item	Gets the view item (element) at the specified ordinal position.
	MaintenanceMode	Gets or sets a value controlling how the view is synchronized with changes in its base data. (Inherited from C1.LiveLinq.LiveViews.View)
	MoveToFirstOnReset	Gets or sets a value indicating that the first item must be made current after initial loading or reset (on any C1.LiveLinq.SourceChangeType.Reset notification) if current item was not set by other means. The default is True. (Inherited from C1.LiveLinq.LiveViews.View)
	Order	Gets a value indicating whether and how this view preserves item order if it exists in its base data source. (Inherited from C1.LiveLinq.LiveViews.View)
	Transaction	Gets an instance of C1.LiveLinq.ITransaction associated with the view. If a view has a transaction associated with it, that transaction's scope is opened automatically every time the view is updated, so the

		programmer does not need to do it manually in code. (Inherited from C1.LiveLinq.LiveViews.View)

Top

See Also

Reference

View<T> Class
C1.LiveLinq.LiveViews Namespace

Indexes Property
Gets the collection of indexes for this view.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Indexes As IndexCollection(Of T)`
C#
`public IndexCollection<T> Indexes {get;}`

Remarks

Live views can be indexed, just like other LiveLinq data sources, to optimize search operations over their data. For an example of an index over a view, see Live Views How To: Create Views Based on Other Views and Create Indexes on Views.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members
Indexes Property

Item Property
The zero-based ordinal position of the view item.

Gets the view item (element) at the specified ordinal position.

Syntax

Visual Basic (Declaration)

`Public ReadOnly Default Property Item( _`
`  ByVal ordinal As Integer _`
`) As T`

C#

`public T this[`
`  int ordinal`
`]; {get;}`

Parameters

ordinal

The zero-based ordinal position of the view item.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members

Events
For a list of all members of this type, see View<T> members.

Public Events

	Name	Description

	Changed	Occurs after an item of the view or the entire view has changed.

Top

See Also

Reference

View<T> Class
C1.LiveLinq.LiveViews Namespace

Changed Event
Occurs after an item of the view or the entire view has changed.

Syntax

Visual Basic (Declaration)

```
Public Event Changed As EventHandler(Of SourceChangeEventArgs(Of T))
```

C#

```
public event EventHandler<SourceChangeEventArgs<T>> Changed
```

Event Data

The event handler receives an argument of type SourceChangeEventArgs<T> containing data related to this event. The following **SourceChangeEventArgs<T>** properties provide information specific to this event.

Property	Description
ChangeType	Gets the type of change.
Item	Gets the object that is being changed.
Ordinal	Gets the ordinal position of the collection item that is being changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

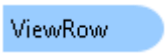
See Also

Reference

View<T> Class
View<T> Members

ViewRow
Represents a view element (item) for the purposes of dynamic, programmatic access to its properties and data binding.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit Class ViewRow` `  Inherits C1.LiveLinq.Collections.IndexableObject`
C#
`public abstract class ViewRow : C1.LiveLinq.Collections.IndexableObject`

Inheritance Hierarchy

System.Object
  C1.LiveLinq.Collections.IndexableObject
    C1.LiveLinq.LiveViews.ViewRow

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

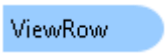
See Also

Reference

ViewRow Members
C1.LiveLinq.LiveViews Namespace
Rows Property

Overview
Represents a view element (item) for the purposes of dynamic, programmatic access to its properties and data binding.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit Class ViewRow` `  Inherits C1.LiveLinq.Collections.IndexableObject`
C#
`public abstract class ViewRow : C1.LiveLinq.Collections.IndexableObject`

Inheritance Hierarchy

System.Object
  C1.LiveLinq.Collections.IndexableObject
    C1.LiveLinq.LiveViews.ViewRow

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRow Members
C1.LiveLinq.LiveViews Namespace
Rows Property

Members

Properties Methods Events

The following tables list the members exposed by ViewRow.

Public Properties

	Name	Description
	Item	Overloaded. Gets or sets a value of the specified view property.
	RowState	Gets the state of a view row with regard to edit, add and delete operations if they are performed directly on the view.
	Tag	Gets or sets user-supplied data associated with the view item.
	Value	Gets the view element (item) represented by this ViewRow object.
	View	Gets the view to which the ViewRow belongs.

Top

Public Methods

	Name	Description
	BeginEdit	Puts the ViewRow into edit mode.
	CancelEdit	Cancels the edit occurring on the row.
	Delete	Deletes a view item.
	EndEdit	Ends the edit occurring on the row.
	ToString	Gets the string representing this view row.

Top

Public Events

	Name	Description
	PropertyChanged	Occurs when a property value changes, after it has been changed.

Top

See Also

Reference

ViewRow Class
C1.LiveLinq.LiveViews Namespace
Rows Property

Methods
For a list of all members of this type, see ViewRow members.

Public Methods

	Name	Description
	BeginEdit	Puts the ViewRow into edit mode.
	CancelEdit	Cancels the edit occurring on the row.
	Delete	Deletes a view item.
	EndEdit	Ends the edit occurring on the row.
	ToString	Gets the string representing this view row.

Top

See Also

Reference

ViewRow Class
C1.LiveLinq.LiveViews Namespace
Rows Property

BeginEdit Method

Puts the ViewRow into edit mode.

Syntax

Visual Basic (Declaration)
`Public Sub BeginEdit()`
C#
`public void BeginEdit()`

Remarks

In edit mode, events and notifications are temporarily suspended, letting the user make changes to more than one property without triggering validation rules.

Edit mode is a standard feature of .NET data binding mechanism, supported by every data source implementing the System.ComponentModel.IEditableObject interface. For detailed explanation, see, for example, System.Data.DataRowView in .NET Framework documentation.

While a view item is in edit mode, changes made to its updatable properties directly in the view are not propagated to the corresponding base data properties until the edit operation is completed by a call to EndEdit (see View.IsReadOnly about updatability of view element properties directly in the view).

If you change base data (source) properties, those changes are propagated to this view and other views depending on that base data according to the normal view maintenance process, regardless of whether the view row is in edit mode or not.

Many-to-one relations between view properties are maintained automatically on changes made to updatable properties directly in the view, regardless of whether the view row is in edit mode or not. For example, if you set a CustomerID directly in the view, CustomerName will change accordingly, even if you do it in edit mode.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRow Class
ViewRow Members

CancelEdit Method
Cancels the edit occurring on the row.

Syntax

Visual Basic (Declaration)
`Public Overridable Sub CancelEdit()`
C#
`public virtual void CancelEdit()`

Remarks

If the user set some of the updatable properties of the row while it was in edit mode, the changes to those properties are rolled back and not propagated to the corresponding base data properties.

See the BeginEdit method for more information.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRow Class
ViewRow Members

Delete Method
Deletes a view item.

Syntax

Visual Basic (Declaration)
`Public Overridable Sub Delete()`

C#

```
public virtual void Delete()
```

Remarks

Deleting a view item is an update operation on the view. As any other update operation performed directly on the view (as opposed to on the base data collection on which that view depends, see C1.LiveLinq.LiveViewExtensions.AsUpdatable<T>), it is allowed only if the view is not read-only (see View.IsReadOnly), and it results in updating (in this case, deleting an item from) one of the view's base data collections.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRow Class
ViewRow Members

EndEdit Method
Ends the edit occurring on the row.

Syntax

Visual Basic (Declaration)

```
Public Overridable Sub EndEdit()
```

C#

```
public virtual void EndEdit()
```

Remarks

If the user set some of the updatable properties of the row while it was in edit mode, the changes to those properties are propagated to the corresponding base data properties at this point.

See the BeginEdit method for more information.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRow Class
ViewRow Members

ToString Method
Gets the string representing this view row.

Syntax

Visual Basic (Declaration)
`Public Overrides Function ToString() As String`
C#
`public override string ToString()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRow Class
ViewRow Members

Properties
For a list of all members of this type, see ViewRow members.

Public Properties

	Name	Description
	Item	Overloaded. Gets or sets a value of the specified view property.
	RowState	Gets the state of a view row with regard to edit, add and delete operations if they are performed directly on the view.
	Tag	Gets or sets user-supplied data associated with the view item.
	Value	Gets the view element (item) represented by this ViewRow object.
	View	Gets the view to which the ViewRow belongs.

Top

See Also

Reference

ViewRow Class
C1.LiveLinq.LiveViews Namespace
Rows Property

Item Property
Gets or sets a value of the specified view property.

Overload List

Overload	Description
Item(Int32)	Gets or sets a value of the specified view property.
Item(String)	Gets or sets a value of the specified view property.

Remarks

Setting a view property is an update operation on the view. As any other update operation performed directly on the view (as opposed to on the base data collection on which that view depends, see C1.LiveLinq.LiveViewExtensions.AsUpdatable<T>), it is allowed only if the view is not read-only (see View.IsReadOnly), and it results in updating one of the view's base data collections.

Only updatable properties can be set. An attempt to set a read-only property results in an exception. See View.IsReadOnly for more details.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRow Class
ViewRow Members

Item(Int32) Property
The ordinal position of the property in the collection of public properties of the type of the view element.

Gets or sets a value of the specified view property.

Syntax

Visual Basic (Declaration)

```
Public Overloads Property Item( _
   ByVal propertyOrdinal As Integer _
) As Object
```

C#

```
public object Item(
   int propertyOrdinal
) {get; set;}
```

Parameters

propertyOrdinal

The ordinal position of the property in the collection of public properties of the type of the view element.

Property Value

The value of the property.

Remarks

Setting a view property is an update operation on the view. As any other update operation performed directly on the view (as opposed to on the base data collection on which that view depends, see C1.LiveLinq.LiveViewExtensions.AsUpdatable<T>), it is allowed only if the view is not read-only (see View.IsReadOnly), and it results in updating one of the view's base data collections.

Only updatable properties can be set. An attempt to set a read-only property results in an exception. See View.IsReadOnly for more details.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRow Class
ViewRow Members
Overload List

Item(String) Property
The name of the property.

Gets or sets a value of the specified view property.

Syntax

Visual Basic (Declaration)

```
Public Overloads Property Item( _
   ByVal propertyName As String _
) As Object
```

C#

```
public object Item(
   string propertyName
```

```
) {get; set;}
```

Parameters

propertyName

The name of the property.

Property Value

The value of the property.

Remarks

Setting a view property is an update operation on the view. As any other update operation performed directly on the view (as opposed to on the base data collection on which that view depends, see C1.LiveLinq.LiveViewExtensions.AsUpdatable<T>), it is allowed only if the view is not read-only (see View.IsReadOnly), and it results in updating one of the view's base data collections.

Only updatable properties can be set. An attempt to set a read-only property results in an exception. See View.IsReadOnly for more details.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRow Class
ViewRow Members
Overload List

RowState Property

Gets the state of a view row with regard to edit, add and delete operations if they are performed directly on the view.

Syntax

Visual Basic (Declaration)	

```
Public ReadOnly Property RowState As ViewRowState
```

C#

```
public ViewRowState RowState {get;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRow Class
ViewRow Members

Tag Property
Gets or sets user-supplied data associated with the view item.

Syntax

Visual Basic (Declaration)

```
Public Property Tag As Object
```

C#

```
public object Tag {get; set;}
```

Remarks

Use this property to store any object you want to associate in your code with the view item that you need to access quickly.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRow Class
ViewRow Members

Value Property
Gets the view element (item) represented by this ViewRow object.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Value As Object`
C#
`public object Value {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRow Class
ViewRow Members

View Property
Gets the view to which the ViewRow belongs.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property View As View`
C#
`public View View {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRow Class
ViewRow Members

Events
For a list of all members of this type, see ViewRow members.

Public Events

	Name	Description
	PropertyChanged	Occurs when a property value changes, after it has been changed.

Top

See Also

Reference

ViewRow Class
C1.LiveLinq.LiveViews Namespace
Rows Property

PropertyChanged Event
Occurs when a property value changes, after it has been changed.

Syntax

Visual Basic (Declaration)
`Public Shadows Event PropertyChanged As PropertyChangedEventHandler`
C#
`public new event PropertyChangedEventHandler PropertyChanged`

Event Data

The event handler receives an argument of type PropertyChangedEventArgs containing data related to this event. The following **PropertyChangedEventArgs** properties provide information specific to this event.

Property	Description
PropertyName	

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

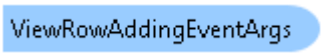
See Also

Reference

ViewRow Class
ViewRow Members

ViewRowAddingEventArgs
Provides data for the ViewRowCollection.ViewRowAdding event.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class ViewRowAddingEventArgs` `  Inherits System.EventArgs`
C#
`public class ViewRowAddingEventArgs : System.EventArgs`

Inheritance Hierarchy

System.Object
  System.EventArgs
    C1.LiveLinq.LiveViews.ViewRowAddingEventArgs

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

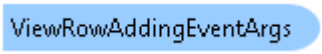
See Also

Reference

ViewRowAddingEventArgs Members
C1.LiveLinq.LiveViews Namespace

Overview
Provides data for the ViewRowCollection.ViewRowAdding event.

Object Model



Syntax

Visual Basic (Declaration)	
`Public Class ViewRowAddingEventArgs` `  Inherits System.EventArgs`	
C#	
`public class ViewRowAddingEventArgs : System.EventArgs`	

Inheritance Hierarchy

System.Object
  System.EventArgs
    C1.LiveLinq.LiveViews.ViewRowAddingEventArgs

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowAddingEventArgs Members
C1.LiveLinq.LiveViews Namespace

Members
Properties

The following tables list the members exposed by ViewRowAddingEventArgs.

Public Properties

	Name	Description
	Row	Gets the new view row that has just been added to ViewRowCollection.

Top

See Also

Reference

ViewRowAddingEventArgs Class
C1.LiveLinq.LiveViews Namespace

Properties
For a list of all members of this type, see ViewRowAddingEventArgs members.

Public Properties

	Name	Description
	Row	Gets the new view row that has just been added to ViewRowCollection.

Top

See Also

Reference

ViewRowAddingEventArgs Class
C1.LiveLinq.LiveViews Namespace

Row Property
Gets the new view row that has just been added to ViewRowCollection.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Row As ViewRow`
C#
`public ViewRow Row {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowAddingEventArgs Class
ViewRowAddingEventArgs Members

ViewRowCollection
Represents a collection of ViewRow objects used for programmatic access to view elements (items) and for data binding.

Object Model

Syntax

Visual Basic (Declaration)

```
Public MustInherit Class ViewRowCollection
   Implements C1.LiveLinq.Indexing.IIndexedSource(Of ViewRow),
C1.LiveLinq.IObservableSource(Of ViewRow)
```

C#

```
public abstract class ViewRowCollection :
C1.LiveLinq.Indexing.IIndexedSource<ViewRow>,
C1.LiveLinq.IObservableSource<ViewRow>
```

Remarks

A **ViewRowCollection** is owned by a view, see View.Rows.

The collection of *view rows* (ViewRow objects) is always synchronized with the collection of view elements.

ViewRow objects provide programmatic access to view elements and their properties. Also, the View.Rows collection serves as the data source for data binding, when you bind a control or another client to a view.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.ViewRowCollection

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Members
C1.LiveLinq.LiveViews Namespace

Overview
Represents a collection of ViewRow objects used for programmatic access to view elements (items) and for data binding.

Object Model

ViewRowCollection

Syntax

Visual Basic (Declaration)	

`Public MustInherit Class ViewRowCollection`
`  Implements C1.LiveLinq.Indexing.IIndexedSource(Of ViewRow),`
`C1.LiveLinq.IObservableSource(Of ViewRow)`	

C#	

`public abstract class ViewRowCollection :`
`C1.LiveLinq.Indexing.IIndexedSource<ViewRow>,`
`C1.LiveLinq.IObservableSource<ViewRow>`	

Remarks

A **ViewRowCollection** is owned by a view, see View.Rows.

The collection of *view rows* (ViewRow objects) is always synchronized with the collection of view elements.

ViewRow objects provide programmatic access to view elements and their properties. Also, the View.Rows collection serves as the data source for data binding, when you bind a control or another client to a view.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.ViewRowCollection

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Members
C1.LiveLinq.LiveViews Namespace

Members
Properties Methods Events

The following tables list the members exposed by ViewRowCollection.

Public Properties

	Name	Description
	AllowClear	Gets a value indicating whether the Clear operation is allowed on the view directly.
	AllowEdit	Gets a value indicating whether modifying property values is allowed on the view directly.
	AllowNew	Gets a value indicating whether the CreateRow operation is allowed on the view directly.
	AllowRemove	Gets a value indicating whether deleting rows is allowed on the view directly.
	Count	Gets the number of elements in the view.
	GroupDescriptions	Gets a collection of System.ComponentModel.GroupDescription objects that describe how the items in the collection are grouped.
	Groups	Gets the top-level groups.
	Indexes	Gets the collection of indexes for this view allowing to search for ViewRow objects.
	Item	Gets the view row at the specified ordinal position.
	Properties	Returns the collection of properties available in the view element type to

		programmatic access through ViewRow and to data binding.
	SortDescriptions	Gets a collection of System.ComponentModel.SortDescription objects that describe how the items in the collection are sorted.

Top

Public Methods

	Name	Description
	Clear	Deletes all view elements.
	Contains	Determines whether the ViewRowCollection contains a specific view row.
	CreateRow	Creates a new item directly in the view, and a view row associated with it, and adds it to the ViewRowCollection.
	DeferRefresh	Enters a defer cycle that you can use to merge changes to the view and delay automatic refresh.
	GetEnumerator	Returns an enumerator that iterates through the collection.
	GetItemProperties	Returns the collection of properties available in the view element type to programmatic access through ViewRow and to data binding.
	IndexOf	Determines the ordinal position of a specific view row in the ViewRowCollection.
	Remove	Deletes the specified view item.
	RemoveAt	Deletes the view row at a specified ordinal position in ViewRowCollection.

Top

Public Events

	Name	Description
	Changed	Occurs after a view row has changed.
	ViewRowAdding	Occurs after a new view row is created so it can be populated with default values.

Top

See Also

Reference

ViewRowCollection Class
C1.LiveLinq.LiveViews Namespace

Methods
For a list of all members of this type, see ViewRowCollection members.

Public Methods

	Name	Description
	Clear	Deletes all view elements.
	Contains	Determines whether the ViewRowCollection contains a specific view row.
	CreateRow	Creates a new item directly in the view, and a view row associated with it, and adds it to the ViewRowCollection.
	DeferRefresh	Enters a defer cycle that you can use to merge changes to the view and delay automatic refresh.
	GetEnumerator	Returns an enumerator that iterates through the collection.
	GetItemProperties	Returns the collection of properties available in the view element type to programmatic access through ViewRow and to data binding.

	IndexOf	Determines the ordinal position of a specific view row in the ViewRowCollection.
	Remove	Deletes the specified view item.
	RemoveAt	Deletes the view row at a specified ordinal position in ViewRowCollection.

Top

See Also

Reference

ViewRowCollection Class
C1.LiveLinq.LiveViews Namespace

Clear Method
Deletes all view elements.

Syntax

Visual Basic (Declaration)
`Public Sub Clear()`
C#
`public void Clear()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

Contains Method

The view row to locate in the collection.

Determines whether the ViewRowCollection contains a specific view row.

Syntax

Visual Basic (Declaration)

`Public Function Contains( _`
`  ByVal row As ViewRow _`
`) As Boolean`

C#

`public bool Contains(`
`  ViewRow row`
`)`

Parameters

row

The view row to locate in the collection.

Return Value

true if the view row is found in the collection; otherwise, **false**.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

CreateRow Method

Creates a new item directly in the view, and a view row associated with it, and adds it to the ViewRowCollection.

Syntax

Visual Basic (Declaration)
`Public Function CreateRow() As ViewRow`
C#
`public ViewRow CreateRow()`

Return Value

The newly created view row.

Remarks

Creating a new row is an update operation on the view. As any other update operation performed directly on the view (as opposed to on the base data collection on which that view depends, see C1.LiveLinq.LiveViewExtensions.AsUpdatable<T>), it is allowed only if the view is not read-only (see View.IsReadOnly), and it results in updating (in this case, adding a new item to) one of the view's base data collections.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

DeferRefresh Method
Enters a defer cycle that you can use to merge changes to the view and delay automatic refresh.

Syntax

Visual Basic (Declaration)
`Public Function DeferRefresh() As IDisposable`

C#

```
public IDisposable DeferRefresh()
```

Return Value

An System.IDisposable object that you can use to dispose of the calling object.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

GetEnumerator Method
Returns an enumerator that iterates through the collection.

Syntax

Visual Basic (Declaration)

```
Public Function GetEnumerator() As IEnumerator(Of ViewRow)
```

C#

```
public IEnumerator<ViewRow> GetEnumerator()
```

Return Value

A IEnumerator that can be used to iterate through the collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

GetItemProperties Method
An array of System.ComponentModel.PropertyDescriptor objects to find in the collection as bindable. Can be null.

Returns the collection of properties available in the view element type to programmatic access through ViewRow and to data binding.

Syntax

Visual Basic (Declaration)

```
Public Function GetItemProperties( _
   ByVal listAccessors() As PropertyDescriptor _
) As PropertyDescriptorCollection
```

C#

```
public PropertyDescriptorCollection GetItemProperties(
   PropertyDescriptor[] listAccessors
)
```

Parameters

listAccessors

An array of System.ComponentModel.PropertyDescriptor objects to find in the collection as bindable. Can be null.

Remarks

If *listAccessors* is null, this method returns the collection of properties of the view element type.

Non-null *listAccessors* is used for obtaining property collection for hierarchical binding: If *listAccessors* contains a single element, then it is used to find an object-valued property in the element type, and the collection of properties of the type of that object-valued property is returned. If *listAccessors* contains two elements, then its second element is used to find an object-valued property in the collection of properties on the previous level, and so on, see ITypedList.GetItemProperties for more information.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

IndexOf Method

The view row to locate in the collection.

Determines the ordinal position of a specific view row in the ViewRowCollection.

Syntax

Visual Basic (Declaration)

```
Public Function IndexOf( _
   ByVal row As ViewRow _
) As Integer
```

C#

```
public int IndexOf(
   ViewRow row
)
```

Parameters

row

The view row to locate in the collection.

Return Value

The ordinal position of the view row in the collection if it is found; otherwise, -1.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

Remove Method
The view row representing the item to delete.

Deletes the specified view item.

Syntax

Visual Basic (Declaration)

```
Public Function Remove( _
   ByVal row As ViewRow _
) As Boolean
```

C#

```
public bool Remove(
   ViewRow row
)
```

Parameters

row

The view row representing the item to delete.

Return Value

true, if the item was deleted as a result of this operation; otherwise, **false**.

Remarks

This is an update operation on the view equivalent to calling ViewRow.Delete.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

RemoveAt Method
The zero-based ordinal position of the item to remove.

Deletes the view row at a specified ordinal position in ViewRowCollection.

Syntax

Visual Basic (Declaration)

```
Public Sub RemoveAt( _
   ByVal ordinal As Integer _
)
```

C#

```
public void RemoveAt(
   int ordinal
)
```

Parameters

ordinal

The zero-based ordinal position of the item to remove.

Remarks

This is an update operation on the view equivalent to calling ViewRow.Delete.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

Properties
For a list of all members of this type, see ViewRowCollection members.

Public Properties

	Name	Description
	AllowClear	Gets a value indicating whether the Clear operation is allowed on the view directly.
	AllowEdit	Gets a value indicating whether modifying property values is allowed on the view directly.
	AllowNew	Gets a value indicating whether the CreateRow operation is allowed on the view directly.
	AllowRemove	Gets a value indicating whether deleting rows is allowed on the view directly.
	Count	Gets the number of elements in the view.
	GroupDescriptions	Gets a collection of System.ComponentModel.GroupDescription objects that describe how the items in the collection are grouped.
	Groups	Gets the top-level groups.
	Indexes	Gets the collection of indexes for this view allowing to search for ViewRow objects.
	Item	Gets the view row at the specified ordinal position.

	Properties	Returns the collection of properties available in the view element type to programmatic access through ViewRow and to data binding.
	SortDescriptions	Gets a collection of System.ComponentModel.SortDescription objects that describe how the items in the collection are sorted.

Top

See Also

Reference

ViewRowCollection Class
C1.LiveLinq.LiveViews Namespace

AllowClear Property
Gets a value indicating whether the Clear operation is allowed on the view directly.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Property AllowClear As Boolean
```

C#

```
public bool AllowClear {get;}
```

Remarks

As any other update operation performed directly on the view (as opposed to on the base data collection on which that view depends, see C1.LiveLinq.LiveViewExtensions.AsUpdatable<T>), it is allowed only if the view is not read-only (see View.IsReadOnly). Note that the same operation is often allowed on the base data collection, and it will normally result in the corresponding change of the view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

AllowEdit Property
Gets a value indicating whether modifying property values is allowed on the view directly.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property AllowEdit As Boolean`
C#
`public bool AllowEdit {get;}`

Remarks

As any other update operation performed directly on the view (as opposed to on the base data collection on which that view depends, see C1.LiveLinq.LiveViewExtensions.AsUpdatable<T>), it is allowed only if the view is not read-only (see View.IsReadOnly).

Although read-only properties of a view cannot be modified directly in the view, they still reflect up-to-date values of the source, so the difference is often not critical, you can always modify corresponding property in the source, that will automatically change the property in the view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

AllowNew Property
Gets a value indicating whether the CreateRow operation is allowed on the view directly.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Property AllowNew As Boolean
```

C#

```
public bool AllowNew {get;}
```

Remarks

As any other update operation performed directly on the view (as opposed to on the base data collection on which that view depends, see C1.LiveLinq.LiveViewExtensions.AsUpdatable<T>), it is allowed only if the view is not read-only (see View.IsReadOnly). Note that the same operation is usually allowed on the base data collection, and it will normally result in the corresponding change of the view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

AllowRemove Property
Gets a value indicating whether deleting rows is allowed on the view directly.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Property AllowRemove As Boolean
```

C#

```
public bool AllowRemove {get;}
```

Remarks

As any other update operation performed directly on the view (as opposed to on the base data collection on which that view depends, see C1.LiveLinq.LiveViewExtensions.AsUpdatable<T>), it is allowed only if the view is not read-only (see View.IsReadOnly). Note that the same operation is often allowed on the base data collection, and it will normally result in the corresponding change of the view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

Count Property
Gets the number of elements in the view.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Count As Integer`
C#
`public int Count {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

GroupDescriptions Property

Gets a collection of System.ComponentModel.GroupDescription objects that describe how the items in the collection are grouped.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property GroupDescriptions As ObservableCollection(Of GroupDescription)`
C#
`public ObservableCollection<GroupDescription> GroupDescriptions {get;}`

Property Value

A collection of System.ComponentModel.GroupDescription objects that describe how the items in the collection are grouped.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

Groups Property

Gets the top-level groups.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Groups As ReadOnlyObservableCollection(Of Object)`
C#

```
public ReadOnlyObservableCollection<object> Groups {get;}
```

Property Value

A read-only collection of the top-level groups.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

Indexes Property
Gets the collection of indexes for this view allowing to search for ViewRow objects.

Syntax

Visual Basic (Declaration)
`Public MustOverride ReadOnly Property Indexes As IndexCollection(Of ViewRow)`
C#
`public abstract IndexCollection<ViewRow> Indexes {get;}`

Remarks

ViewRowCollection can be indexed, just like other LiveLinq data sources, to optimize searches for ViewRow objects. It implements the C1.LiveLinq.Indexing.IIndexedSource<T> interface.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members
Indexes Property

Item Property
The zero-based ordinal position of the view row.

Gets the view row at the specified ordinal position.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Default Property Item( _
   ByVal ordinal As Integer _
) As ViewRow
```

C#

```
public ViewRow this[
   int ordinal
]; {get;}
```

Parameters

ordinal

The zero-based ordinal position of the view row.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

Properties Property

Returns the collection of properties available in the view element type to programmatic access through ViewRow and to data binding.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Properties As ViewRowPropertyInfo()`
C#
`public ViewRowPropertyInfo[] Properties {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

SortDescriptions Property

Gets a collection of System.ComponentModel.SortDescription objects that describe how the items in the collection are sorted.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property SortDescriptions As SortDescriptionCollection`
C#
`public SortDescriptionCollection SortDescriptions {get;}`

Property Value

A collection of System.ComponentModel.SortDescription objects that describe how the items in the collection are sorted.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

Events
For a list of all members of this type, see ViewRowCollection members.

Public Events

	Name	Description
	Changed	Occurs after a view row has changed.
	ViewRowAdding	Occurs after a new view row is created so it can be populated with default values.

Top

See Also

Reference

ViewRowCollection Class
C1.LiveLinq.LiveViews Namespace

Changed Event
Occurs after a view row has changed.

Syntax

Visual Basic (Declaration)
`Public Event Changed As EventHandler(Of SourceChangeEventArgs(Of ViewRow))`

C#	
`public event EventHandler<SourceChangeEventArgs<ViewRow>> Changed`	

Event Data

The event handler receives an argument of type SourceChangeEventArgs<T> containing data related to this event. The following **SourceChangeEventArgs<T>** properties provide information specific to this event.

Property	Description
ChangeType	Gets the type of change.
Item	Gets the object that is being changed.
Ordinal	Gets the ordinal position of the collection item that is being changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

ViewRowAdding Event
Occurs after a new view row is created so it can be populated with default values.

Syntax

Visual Basic (Declaration)	
`Public Event ViewRowAdding As EventHandler(Of ViewRowAddingEventArgs)`	
C#	

```
public event EventHandler<ViewRowAddingEventArgs> ViewRowAdding
```

Event Data

The event handler receives an argument of type ViewRowAddingEventArgs containing data related to this event. The following **ViewRowAddingEventArgs** properties provide information specific to this event.

Property	Description
Row	Gets the new view row that has just been added to ViewRowCollection.

Remarks

This event occurs only if the new row is created directly in the view, as a result of a view update operation, that is, with CreateRow or via data binding. It does not occur when new rows appear in the view as a result of view maintenance on changes made to the view's source data collections.

This event occurs immediately on creating the new row, before the method creating it returns, before the row enters edit mode (see ViewRowState about edit mode).

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

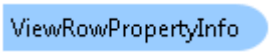
See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

ViewRowPropertyInfo
Allows to control certain behavior of a property of the element type of a View.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class ViewRowPropertyInfo`
C#
`public class ViewRowPropertyInfo`

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.ViewRowPropertyInfo

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

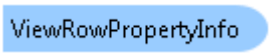
See Also

Reference

ViewRowPropertyInfo Members
C1.LiveLinq.LiveViews Namespace

Overview
Allows to control certain behavior of a property of the element type of a View.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class ViewRowPropertyInfo`
C#
`public class ViewRowPropertyInfo`

Inheritance Hierarchy

System.Object
C1.LiveLinq.LiveViews.ViewRowPropertyInfo

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowPropertyInfo Members
C1.LiveLinq.LiveViews Namespace

Members
Properties

The following tables list the members exposed by ViewRowPropertyInfo.

Public Properties

	Name	Description
	ImmediateUpdate	Gets or sets a boolean value indicating whether changes made to this property through data binding must be immediately sent to the corresponding view item even if the view in editing mode.
	PropertyName	Gets the name of the property.

Top

See Also

Reference

ViewRowPropertyInfo Class
C1.LiveLinq.LiveViews Namespace

Properties
>

Name	Description
ImmediateUpdate	Gets or sets a boolean value indicating whether changes made to this property through data binding must be immediately sent to the corresponding view item even if the view in editing mode.
PropertyName	Gets the name of the property.

Top

See Also

Reference

ViewRowPropertyInfo Class
C1.LiveLinq.LiveViews Namespace

ImmediateUpdate Property
Gets or sets a boolean value indicating whether changes made to this property through data binding must be immediately sent to the corresponding view item even if the view in editing mode.

Syntax

Visual Basic (Declaration)

```
Public Property ImmediateUpdate As Boolean
```

C#

```
public bool ImmediateUpdate {get; set;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowPropertyInfo Class
ViewRowPropertyInfo Members
BeginEdit Method

PropertyName Property
Gets the name of the property.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property PropertyName As String`
C#
`public string PropertyName {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowPropertyInfo Class
ViewRowPropertyInfo Members

Enumerations

DataBindingMode
Enumeration of the possible data binding modes. It is used by the View.DataBindingMode property.

Syntax

Visual Basic (Declaration)
`Public Enum DataBindingMode` `  Inherits System.Enum`
C#
`public enum DataBindingMode : System.Enum`

Members

Member	Description
Default	The default mode, which is WPF mode in WPF applications and WinForms mode in WinForms applications.
WinForms	In data binding to a View, view rows (elements of the View.Rows collection) are used for data binding.
WPF	In data binding to a View, view items (elements of the View itself as a collection) are used for data binding.

Inheritance Hierarchy

System.Object
  System.ValueType
    System.Enum
      C1.LiveLinq.LiveViews.DataBindingMode

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1.LiveLinq.LiveViews Namespace

ViewMaintenanceMode
Specifies how a view is synchronized with changes in its base data.

Syntax

Visual Basic (Declaration)
`Public Enum ViewMaintenanceMode` `  Inherits System.Enum`

C#
`public enum ViewMaintenanceMode : System.Enum`

Members

Member	Description
Default	By default, the view is in a "smart mode": It is in **Deferred** mode if nobody is interested in its data, and it is synchronized and goes to **Immediate** mode if there is a client interested in receiving notifications of changes in that view's data. In other words, a view in **Default** mode is effectively in **Deferred** mode if no listeners registered to be notified of its changes, and it is effectively in **Immediate** mode if there are such listeners (for example, if there is GUI control bound to it).
Deferred	When a change in base data occur, the view is not synchronized with it immediately. It is allowed to go stale, out of sync with its base data as long as there are no requests for this view's data. The view is synchronized on demand, that is, it is synchronized when any request for its data arrives.
Immediate	The view is synchronized with its base data automatically and immediately after any change in its base data occurs. The view is kept synchronized with its base data at all times.

Remarks

See Also: View Maintenance Mode.

Inheritance Hierarchy

System.Object
  System.ValueType
    System.Enum
      C1.LiveLinq.LiveViews.ViewMaintenanceMode

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1.LiveLinq.LiveViews Namespace
MaintenanceMode Property
DeferredMaintenance Property

ViewOrder
Specifies whether and how a view must preserve item order if it exists in the source.

Syntax

Visual Basic (Declaration)
`Public Enum ViewOrder` `  Inherits System.Enum`
C#
`public enum ViewOrder : System.Enum`

Members

Member	Description
NotPreserved	Source order is not preserved. Preserving source order is not guaranteed even at view creation.
PartiallyPreserved	Source order is partially preserved. When the view is created, it preserves source order, but later, when changes occur in the source, view items added or modified to reflect those changes aren't guaranteed to appear at the same order position in the view as in the source.
Preserved	Source order is preserved completely. Order of items in the view is always the same as in the source (if source has a particular order), even after the view is

	maintained to reflect changes that occurred in the source.

Inheritance Hierarchy

System.Object
  System.ValueType
    System.Enum
      C1.LiveLinq.LiveViews.ViewOrder

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1.LiveLinq.LiveViews Namespace

ViewRowState
The state of a view row with regard to edit, add and delete operations if they are performed directly on the view.

Syntax

Visual Basic (Declaration)
`Public Enum ViewRowState` `  Inherits System.Enum`
C#
`public enum ViewRowState : System.Enum`

Members

Member	Description
Detached	The row was deleted, or it is a new row after exiting edit mode.

Modified	The row is in edit mode (ViewRow.BeginEdit was called, neither ViewRow.EndEdit nor ViewRow.CancelEdit was called yet), and the row is not new (was not created by ViewRowCollection.CreateRow).
New	The row was added to the view directly (not by adding to a basic data collection) by calling ViewRowCollection.CreateRow or through data binding (such new row enters edit mode once it is created) and still in edit mode (neither ViewRow.EndEdit nor ViewRow.CancelEdit was called yet).
Unmodified	The row is a regular view row, not in edit mode and not deleted.

Remarks

This state concerns edit, add and delete operations performed directly on the view (programmatically via ViewRow objects or through data binding). It does not concern modifications made to the view's base (source) data collections. Modifications made to source data can also change view items, as a result of the normal view maintenance process, see View.MaintenanceMode, but in that case the state of thus added or modified rows remains **Unmodified**.

Note on adding rows directly to the view:

If a row is added directly to the view (as opposed to adding it to one of its base data collections), the following happens:

When a new row is created with ViewRowCollection.CreateRow or through data binding, it enters edit mode.

When it is committed with ViewRow.EndEdit, a new row is added to the view's base data collection, and, usually, a corresponding row appears in the view, that has **Unmodified** state, although in some cases it may be more than one row or none, depending on the view query. The original view row no longer corresponds to a view item after the ViewRow.EndEdit or ViewRow.CancelEdit call, its state becomes **Detached**. Accessing it after that would throw an exception.

Inheritance Hierarchy

System.Object
  System.ValueType
    System.Enum
      C1.LiveLinq.LiveViews.ViewRowState

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1.LiveLinq.LiveViews Namespace

C1.LiveLinq.LiveViews.Xml Namespace

Overview

Classes

	Class	Description
	XHint	The class used for the return type in the XmlExtensions.IndexedElement and XmlExtensions.IndexedAttribute extension methods.
	XmlExtensions	Provides a set of static (extension) methods for LiveLinq to XML.

See Also

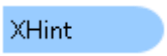
Reference

C1.LiveLinq.4 Assembly

Classes

XHint
The class used for the return type in the XmlExtensions.IndexedElement and XmlExtensions.IndexedAttribute extension methods.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit Class XHint`
C#
`public abstract class XHint`

Remarks

This class is not used in execution, because hints like XmlExtensions.IndexedElement and XmlExtensions.IndexedAttribute are never executed, they are used merely to convey information ('hint') to LiveLinq query optimizer, they are discarded before the query is executed. The only purpose of this class is to preserve syntactical correctness of the query.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.Xml.XHint

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

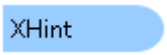
See Also

Reference

XHint Members
C1.LiveLinq.LiveViews.Xml Namespace

Overview
The class used for the return type in the XmlExtensions.IndexedElement and XmlExtensions.IndexedAttribute extension methods.

Object Model



Syntax

Visual Basic (Declaration)	
`Public MustInherit Class XHint`	
C#	
`public abstract class XHint`	

Remarks

This class is not used in execution, because hints like XmlExtensions.IndexedElement and XmlExtensions.IndexedAttribute are never executed, they are used merely to convey information ('hint') to LiveLinq query optimizer, they are discarded before the query is executed. The only purpose of this class is to preserve syntactical correctness of the query.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.Xml.XHint

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XHint Members
C1.LiveLinq.LiveViews.Xml Namespace

Members
The following tables list the members exposed by XHint.

Public Operators

Explicit Type Conversion	Overloaded.

Top

See Also

Reference

XHint Class
C1.LiveLinq.LiveViews.Xml Namespace

Operators

Explicit Type Conversion Operator

Overload List

Overload	Description
Explicit Type Conversion(String,XHint)	
Explicit Type Conversion(Boolean,XHint)	
Explicit Type Conversion(Int32,XHint)	
Explicit Type Conversion(UInt32,XHint)	
Explicit Type Conversion(Int64,XHint)	
Explicit Type Conversion(UInt64,XHint)	
Explicit Type Conversion(Single,XHint)	
Explicit Type Conversion(Double,XHint)	
Explicit Type Conversion(Decimal,XHint)	
Explicit Type Conversion(DateTime,XHint)	
Explicit Type Conversion(DateTimeOffset,XHint)	
Explicit Type Conversion(TimeSpan,XHint)	
Explicit Type Conversion(Guid,XHint)	

Explicit Type Conversion(Nullable<Boolean>,XHint)	
Explicit Type Conversion(Nullable<Int32>,XHint)	
Explicit Type Conversion(Nullable<UInt32>,XHint)	
Explicit Type Conversion(Nullable<Int64>,XHint)	
Explicit Type Conversion(Nullable<UInt64>,XHint)	
Explicit Type Conversion(Nullable<Single>,XHint)	
Explicit Type Conversion(Nullable<Double>,XHint)	
Explicit Type Conversion(Nullable<Decimal>,XHint)	
Explicit Type Conversion(Nullable<DateTime>,XHint)	
Explicit Type Conversion(Nullable<DateTimeOffset>,XHint)	
Explicit Type Conversion(Nullable<TimeSpan>,XHint)	
Explicit Type Conversion(Nullable<Guid>,XHint)	

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

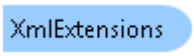
See Also

Reference

XHint Class
XHint Members

XmlExtensions
Provides a set of static (extension) methods for LiveLinq to XML.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit NotInheritable Class XmlExtensions`
C#
`public static class XmlExtensions`

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.Xml.XmlExtensions

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

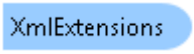
See Also

Reference

XmlExtensions Members
C1.LiveLinq.LiveViews.Xml Namespace

Overview
Provides a set of static (extension) methods for LiveLinq to XML.

Object Model



Syntax

Visual Basic (Declaration)

```
Public MustInherit NotInheritable Class XmlExtensions
```

C#

```
public static class XmlExtensions
```

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.Xml.XmlExtensions

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Members
C1.LiveLinq.LiveViews.Xml Namespace

Members
Methods

The following tables list the members exposed by XmlExtensions.

Public Methods

	Name	Description
S	AsLive	Overloaded. Creates a view based on the specified XML document.
S	Attributes	Overloaded. Returns a view representing the collection of the attributes of every element in the source view.
S	BeginUpdate	Suspends notifications while massive changes are being made to an XML node and its descendants.

S	DescendantNodes<T>	Returns a view representing the collection of the descendent nodes of every element and document in the source view.
S	DescendantNodesAndSelf	Returns a view representing the collection of nodes that contains every element in the source view, and the descendent nodes of every element in the source view.
S	Descendants	Overloaded. Returns a view representing the collection of elements that contains the descendent elements of every element and document in the source view.
S	DescendantsAndSelf	Overloaded. Returns a view representing a collection of elements that contains every element in the source view, and the descendent elements of every element in the source view.
S	Elements	Overloaded. Returns a view representing the collection of child elements of every element and document in the source view..
S	EndUpdate	Ends notification suspension started with BeginUpdate.
S	IndexedAttribute	Overloaded. A hint to create and use an index on the specified XML attribute. The hint has default action.
S	IndexedElement	Overloaded. A hint to create and use an index on the specified XML element. The hint has default action.
S	Nodes<T>	Returns a view representing the collection of child nodes of every document and element in the source view.
S	Root	Gets the view for the root element of the XML tree for this document.

Top

See Also

Reference

XmlExtensions Class
C1.LiveLinq.LiveViews.Xml Namespace

Methods
>

Name	Description
AsLive	Overloaded. Creates a view based on the specified XML document.
Attributes	Overloaded. Returns a view representing the collection of the attributes of every element in the source view.
BeginUpdate	Suspends notifications while massive changes are being made to an XML node and its descendants.
DescendantNodes<T>	Returns a view representing the collection of the descendent nodes of every element and document in the source view.
DescendantNodesAndSelf	Returns a view representing the collection of nodes that contains every element in the source view, and the descendent nodes of every element in the source view.
Descendants	Overloaded. Returns a view representing the collection of elements that contains the descendent elements of every element and document in the source view.
DescendantsAndSelf	Overloaded. Returns a view representing a collection of elements that contains every element in the source view, and the descendent elements of every element in the source view.
Elements	Overloaded. Returns a view representing the collection of child elements of every element and document in the source view..
EndUpdate	Ends notification suspension started with BeginUpdate.
IndexedAttribute	Overloaded. A hint to create and use an index on the specified XML attribute. The hint has default action.
IndexedElement	Overloaded. A hint to create and use an index on the specified XML element. The hint has default action.

Nodes<T>	Returns a view representing the collection of child nodes of every document and element in the source view.
Root	Gets the view for the root element of the XML tree for this document.

Top

See Also

Reference

XmlExtensions Class
C1.LiveLinq.LiveViews.Xml Namespace

AsLive Method
Creates a view based on the specified XML document.

Overload List

Overload	Description
AsLive(XDocument)	Creates a view based on the specified XML document.
AsLive(XElement)	Creates a view based on the specified XML element.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members

AsLive(XDocument) Method
The XML document to expose as a view.

Creates a view based on the specified XML document.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function AsLive( _`
`  ByVal document As XDocument _`
`) As View(Of XDocument)`

C#

`public static View<XDocument> AsLive(`
`  XDocument document`
`)`

Parameters

document

The XML document to expose as a view.

Return Value

A view representing the specified XML document.

Remarks

Since there can be only one root element in a document, the view based on a document (`document.AsLive()`) is similar to the view based on its root element (`document.Root.AsLive()`). The difference is that the document view contains the document as its only item, and the root view contains the root as its only item. This difference is essential only when the root of the document is replaced with a different element (and so the whole XML tree changes), then the document view remains valid and shows the changed document contents, whereas the root view becomes disconnected from the document.

Note: This view is owned by the System.Xml.Linq.XDocument object (it is stored as one of its annotations), so, if you create a view for the same document several times, it will be the same object.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members
Overload List

AsLive(XElement) Method
The XML element to expose as a view.

Creates a view based on the specified XML element.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function AsLive( _
   ByVal element As XElement _
) As View(Of XElement)
```

C#

```
public static View<XElement> AsLive(
   XElement element
)
```

Parameters

element

The XML element to expose as a view.

Return Value

A view representing the specified XML element.

Remarks

This view represents a single XML node. Therefore, as a collection of items, this view's **Count** is always 1. This view is usually used as a starting point to construct a view containing elements or attributes from this node's descendants by using a query with operators from XmlExtensions such as **Elements**, **Descendants** and others, in combination with standard LINQ query operators **where**, **join** and others.

Note: This view is owned by the System.Xml.Linq.XElement object (it is stored as one of its annotations), so, if you create a view for the same element several times, it will be the same object.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members
Overload List

Attributes Method
Returns a view representing the collection of the attributes of every element in the source view.

Overload List

Overload	Description
Attributes(View<XElement>)	Returns a view representing the collection of the attributes of every element in the source view.
Attributes(View<XElement>,XName)	Returns a view representing a filtered collection of the attributes of every element in the source view. Only attributes that have a matching System.Xml.Linq.XName are included.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members

Attributes(View<XElement>) Method
The source view.

Returns a view representing the collection of the attributes of every element in the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function Attributes( _
   ByVal view As View(Of XElement) _
) As View(Of XAttribute)
```

C#

```
public static View<XAttribute> Attributes(
   View<XElement> view
)
```

Parameters

view

The source view.

Return Value

A view containing the attributes of every element in the source view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members
Overload List

The source view.

The System.Xml.Linq.XName to match.

Returns a view representing a filtered collection of the attributes of every element in the source view. Only attributes that have a matching System.Xml.Linq.XName are included.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function Attributes( _
   ByVal view As View(Of XElement), _
   ByVal name As XName _
) As View(Of XAttribute)
```

C#

```
public static View<XAttribute> Attributes(
   View<XElement> view,
   XName name
)
```

Parameters

view

The source view.

name

The System.Xml.Linq.XName to match.

Return Value

A view that contains a filtered collection of the attributes of every element in the source view. Only attributes that have a matching System.Xml.Linq.XName are included.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members
Overload List

BeginUpdate Method
The node that is the root of a tree where massive changes are made in code.

Suspends notifications while massive changes are being made to an XML node and its descendants.

Syntax

Visual Basic (Declaration)

```
Public Shared Sub BeginUpdate( _
   ByVal node As XContainer _
)
```

C#

```
public static void BeginUpdate(
   XContainer node
)
```

Parameters

node

The node that is the root of a tree where massive changes are made in code.

Remarks

This method must be followed by EndUpdate.

Use this method when you already have indexes over this XML or live views based on it, and you need to perform massive changes on the contents of this node and its descendants. Without this method, every single change you make causes LiveLinq to perform necessary operations for maintaining your indexes and live views dependent on this node and its descendants. In case of massive changes, this can be slower than to wait until the massive changes are done and rebuild the indexes and live views.

Between **BeginUpdate** and EndUpdate calls, indexes, live views, bound controls and other change notification listeners are not updated, they don't receive change notifications. When EndUpdate is called, a SourceChangeType.Reset notification is sent, meaning all indexes, live views and other collections dependent on this node and its descendants must be rebuilt from scratch.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members

DescendantNodes<T> Method
The type of the objects in the source *view*, constrained to System.Xml.Linq.XContainer.

The source view.

Returns a view representing the collection of the descendent nodes of every element and document in the source view.

Syntax

Visual Basic (Declaration)

```
Public Shared Function DescendantNodes(Of T As XContainer)( _
   ByVal view As View(Of T) _
) As View(Of XNode)
```

C#

```
public static View<XNode> DescendantNodes<T>(
   View<T> view
)
where T: XContainer
```

Parameters

view

The source view.

Type Parameters

T

The type of the objects in the source *view*, constrained to System.Xml.Linq.XContainer.

Return Value

A view containing every descendent node of every document and element in the source view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members

DescendantNodesAndSelf Method
The source view.

Returns a view representing the collection of nodes that contains every element in the source view, and the descendent nodes of every element in the source view.

Syntax

Visual Basic (Declaration)

```
Public Shared Function DescendantNodesAndSelf( _
   ByVal view As View(Of XElement) _
) As View(Of XNode)
```

C#

```
public static View<XNode> DescendantNodesAndSelf(
   View<XElement> view
)
```

Parameters

view

The source view.

Return Value

A view containing every element in the source view, and the descendent nodes of every element in the source view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members

Descendants Method
Returns a view representing the collection of elements that contains the descendent elements of every element and document in the source view.

Overload List

Overload	Description
Descendants<T>(View<T>)	Returns a view representing the collection of elements that contains the descendent elements of every element and document in the source view.
Descendants<T>(View<T>,XName)	Returns a view representing a filtered collection of elements that contains the descendent elements of every element and document in the source view. Only elements that have a matching System.Xml.Linq.XName are included.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members

Descendants<T>(View<T>) Method
The type of the objects in the source *view*, constrained to System.Xml.Linq.XContainer.

The source view.

Returns a view representing the collection of elements that contains the descendent elements of every element and document in the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function Descendants(Of T As XContainer)( _
   ByVal view As View(Of T) _
) As View(Of XElement)
```

C#

```
public static View<XElement> Descendants<T>(
   View<T> view
)
where T: XContainer
```

Parameters

view

The source view.

Type Parameters

T

The type of the objects in the source *view*, constrained to System.Xml.Linq.XContainer.

Return Value

A view containing every descendent element of every element and document in the source view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members
Overload List

Descendants<T>(View<T>,XName) Method
The type of the objects in the source *view*, constrained to System.Xml.Linq.XContainer.

The source view.

The System.Xml.Linq.XName to match.

Returns a view representing a filtered collection of elements that contains the descendent elements of every element and document in the source view. Only elements that have a matching System.Xml.Linq.XName are included.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function Descendants(Of T As XContainer)( _
   ByVal view As View(Of T), _
   ByVal name As XName _
) As View(Of XElement)
```

C#

```
public static View<XElement> Descendants<T>(
   View<T> view,
   XName name
```

```
)
where T: XContainer
```

Parameters

view

The source view.

name

The System.Xml.Linq.XName to match.

Type Parameters

T

The type of the objects in the source *view*, constrained to System.Xml.Linq.XContainer.

Return Value

A view containing descendants of elements and documents in the source view. Only elements that have a matching System.Xml.Linq.XName are included.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members
Overload List

DescendantsAndSelf Method

Returns a view representing a collection of elements that contains every element in the source view, and the descendent elements of every element in the source view.

Overload List

Overload	Description

DescendantsAndSelf(View<XElement>)	Returns a view representing a collection of elements that contains every element in the source view, and the descendent elements of every element in the source view.
DescendantsAndSelf(View<XElement>,XName)	Returns a view representing a filtered collection of elements that contains every element in the source view, and the descendants of every element in the source view. Only elements that have a matching System.Xml.Linq.XName are included.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members

DescendantsAndSelf(View<XElement>) Method
The source view.

Returns a view representing a collection of elements that contains every element in the source view, and the descendent elements of every element in the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function DescendantsAndSelf( _
   ByVal view As View(Of XElement) _
) As View(Of XElement)
```

C#

```
public static View<XElement> DescendantsAndSelf(
```

```
   View<XElement> view
)
```

Parameters

view

The source view.

Return Value

A view containing every element in the source view, and the descendent elements of every element in the source view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members
Overload List

DescendantsAndSelf(View<XElement>,XName) Method
The source view.

The System.Xml.Linq.XName to match.

Returns a view representing a filtered collection of elements that contains every element in the source view, and the descendants of every element in the source view. Only elements that have a matching System.Xml.Linq.XName are included.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function DescendantsAndSelf( _
   ByVal view As View(Of XElement), _
   ByVal name As XName _
) As View(Of XElement)
```

C#

```
public static View<XElement> DescendantsAndSelf(
   View<XElement> view,
   XName name
)
```

Parameters

view

The source view.

name

The System.Xml.Linq.XName to match.

Return Value

A view containing elements in the source view, and the descendants of elements in the source view. Only elements that have a matching System.Xml.Linq.XName are included.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members
Overload List

Elements Method
Returns a view representing the collection of child elements of every element and document in the source view..

Overload List

Overload	Description

Elements<T>(View<T>)	Returns a view representing the collection of child elements of every element and document in the source view..
Elements<T>(View<T>,XName)	Returns a view representing filtered collection of child elements of every element and document in the source view. Only elements that have a matching System.Xml.Linq.XName are included.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members

Elements<T>(View<T>) Method
The type of the objects in the source *view*, constrained to System.Xml.Linq.XContainer.

The source view.

Returns a view representing the collection of child elements of every element and document in the source view..

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function Elements(Of T As XContainer)( _
   ByVal view As View(Of T) _
) As View(Of XElement)
```

C#

```
public static View<XElement> Elements<T>(
   View<T> view
)
where T: XContainer
```

Parameters

view

The source view.

Type Parameters

T

The type of the objects in the source *view*, constrained to System.Xml.Linq.XContainer.

Return Value

A view containing the child elements of every element and document in the source view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members
Overload List

Elements<T>(View<T>,XName) Method
The type of the objects in the source *view*, constrained to System.Xml.Linq.XContainer.

The source view.

The System.Xml.Linq.XName to match.

Returns a view representing filtered collection of child elements of every element and document in the source view. Only elements that have a matching System.Xml.Linq.XName are included.

Syntax

Visual Basic (Declaration)	
`Public Overloads Shared Function Elements(Of T As XContainer)( _` `  ByVal view As View(Of T), _`	

```
   ByVal name As XName _
) As View(Of XElement)
```

C#

```
public static View<XElement> Elements<T>(
   View<T> view,
   XName name
)
where T: XContainer
```

Parameters

view

The source view.

name

The System.Xml.Linq.XName to match.

Type Parameters

T

The type of the objects in the source *view*, constrained to System.Xml.Linq.XContainer.

Return Value

A view that contains a filtered collection of child elements of every element and document in the source view. Only elements that have a matching System.Xml.Linq.XName are included.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members
Overload List

EndUpdate Method

The node that is the root of a tree where massive changes have been made since the BeginUpdate call.

Ends notification suspension started with BeginUpdate.

Syntax

Visual Basic (Declaration)

`Public Shared Sub EndUpdate( _`
`  ByVal node As XContainer _`
`)`

C#

`public static void EndUpdate(`
`  XContainer node`
`)`

Parameters

node

The node that is the root of a tree where massive changes have been made since the BeginUpdate call.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members

IndexedAttribute Method

A hint to create and use an index on the specified XML attribute. The hint has default action.

Overload List

Overload	Description
IndexedAttribute(XElement,XName)	A hint to create and use an index on the specified XML attribute. The hint has default action.
IndexedAttribute(XAttribute)	A hint to create and use an index on the specified XML attribute. The hint has default action.
IndexedAttribute(XAttribute,IndexingHintAction)	A hint to create and use an index on the specified XML attribute. The hint has specified action.
IndexedAttribute(XElement,XName,IndexingHintAction)	A hint to create and use an index on the specified XML attribute. The hint has specified action.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members

IndexedAttribute(XElement,XName) Method
Example

The element containing the attribute.

The name of the attribute to get.

A hint to create and use an index on the specified XML attribute. The hint has default action.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function IndexedAttribute( _
   ByVal element As XElement, _
   ByVal name As XName _
) As XHint
```

C#

```
public static XHint IndexedAttribute(
   XElement element,
   XName name
)
```

Parameters

element

The element containing the attribute.

name

The name of the attribute to get.

Return Value

The attribute that has the specified name.

Remarks

Hints are used declaratively. They tell LiveLinq query optimizer to create and use an index on this attribute, if possible. When the query is executed, the hint method **IndexedAttribute** is replaced with the standard LINQ to XML method System.Xml.Linq.XElement.Attribute(System.Xml.Linq.XName). See C1.LiveLinq.Hints for more details.

Example

- C#

```
var query =
    from c in customers
    where (string)c.IndexedAttribute("CustomerID") == "ALFKI"
    select c;
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members
Overload List

IndexedAttribute(XAttribute) Method
Example

The attribute to apply the hint to.

A hint to create and use an index on the specified XML attribute. The hint has default action.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function IndexedAttribute( _`
`  ByVal attribute As XAttribute _`
`) As XHint`

C#

`public static XHint IndexedAttribute(`
`  XAttribute attribute`
`)`

Parameters

attribute

The attribute to apply the hint to.

Return Value

Formally, the hint returns the same value that it receives in the parameter. In fact, it is never executed, its role is purely declarative.

Remarks

Hints are used declaratively. They tell LiveLinq query optimizer to create and use an index on this attribute, if possible. After it is used for query optimization, before the query is executed, this hint is removed from the expression. See C1.LiveLinq.Hints for more details.

Example

- C#

```
var query =
    from c in customers
    where (string)c.Attribute("CustomerID").IndexedAttribute() == "ALFKI"
    select c;
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members
Overload List

IndexedAttribute(XAttribute,IndexingHintAction) Method
Example

The attribute to apply the hint to.

The action specified by the hint.

A hint to create and use an index on the specified XML attribute. The hint has specified action.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function IndexedAttribute( _
   ByVal attribute As XAttribute, _
   ByVal action As IndexingHintAction _
) As XHint
```

C#

```
public static XHint IndexedAttribute(
   XAttribute attribute,
   IndexingHintAction action
)
```

Parameters

attribute

The attribute to apply the hint to.

action

The action specified by the hint.

Return Value

Formally, the hint returns the same value that it receives in the parameter. In fact, it is never executed, its role is purely declarative.

Remarks

Hints are used declaratively. They tell LiveLinq query optimizer to create and use an index on this attribute, if possible. After it is used for query optimization, before the query is executed, this hint is removed from the expression. See C1.LiveLinq.Hints for more details.

Example

- C#

```
var query =
    from c in customers
    where
(string)c.Attribute("CustomerID").IndexedAttribute(IndexingHintAction.M
andatory) == "ALFKI"
    select c;
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members
Overload List

IndexedAttribute(XElement,XName,IndexingHintAction) Method
Example

The element containing the attribute.

The name of the attribute to get.

The action specified by the hint.

A hint to create and use an index on the specified XML attribute. The hint has specified action.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function IndexedAttribute( _
   ByVal element As XElement, _
   ByVal name As XName, _
   ByVal action As IndexingHintAction _
) As XHint
```

C#

```
public static XHint IndexedAttribute(
   XElement element,
   XName name,
   IndexingHintAction action
)
```

Parameters

element

The element containing the attribute.

name

The name of the attribute to get.

action

The action specified by the hint.

Return Value

The attribute that has the specified name.

Remarks

Hints are used declaratively. They tell LiveLinq query optimizer to create and use an index on this attribute, if possible. When the query is executed, the hint method **IndexedAttribute** is replaced with the standard LINQ to XML method System.Xml.Linq.XElement.Attribute(System.Xml.Linq.XName). See C1.LiveLinq.Hints for more details.

Example

- C#

```
var query =
    from c in customers
    where (string)c.IndexedAttribute("CustomerID",
IndexingHintAction.Mandatory) == "ALFKI"
    select c;
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members
Overload List

IndexedElement Method

A hint to create and use an index on the specified XML element. The hint has default action.

Overload List

Overload	Description

IndexedElement(XContainer,XName)	A hint to create and use an index on the specified XML element. The hint has default action.
IndexedElement(XElement)	A hint to create and use an index on the specified XML element. The hint has default action.
IndexedElement(XElement,IndexingHintAction)	A hint to create and use an index on the specified XML element. The hint has specified action.
IndexedElement(XContainer,XName,IndexingHintAction)	A hint to create and use an index on the specified XML element. The hint has specified action.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members

IndexedElement(XContainer,XName) Method
Example

The element or document containing the element.

The name of the element to get.

A hint to create and use an index on the specified XML element. The hint has default action.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function IndexedElement( _
   ByVal container As XContainer, _
   ByVal name As XName _
) As XHint
```

C#

```
public static XHint IndexedElement(
   XContainer container,
   XName name
)
```

Parameters

container

The element or document containing the element.

name

The name of the element to get.

Return Value

The element that has the specified name.

Remarks

Hints are used declaratively. They tell LiveLinq query optimizer to create and use an index on this element, if possible. When the query is executed, the hint method **IndexedElement** is replaced with the standard LINQ to XML method System.Xml.Linq.XContainer.Element(System.Xml.Linq.XName). See C1.LiveLinq.Hints for more details.

Example

- C#

```
var query =
    from c in customers
    where (string)c.IndexedElement("CustomerID") == "ALFKI"
    select c;
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members
Overload List

IndexedElement(XElement) Method
Example

The element to apply the hint to.

A hint to create and use an index on the specified XML element. The hint has default action.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function IndexedElement( _`
`  ByVal element As XElement _`
`) As XHint`

C#

`public static XHint IndexedElement(`
`  XElement element`
`)`

Parameters

element

The element to apply the hint to.

Return Value

Formally, the hint returns the same value that it receives in the parameter. In fact, it is never executed, its role is purely declarative.

Remarks

Hints are used declaratively. They tell LiveLinq query optimizer to create and use an index on this element, if possible. After it is used for query optimization, before the query is executed, this hint is removed from the expression. See C1.LiveLinq.Hints for more details.

Example

- C#

```
var query =
    from c in customers
    where (string)c.Element("CustomerID").IndexedElement() == "ALFKI"
    select c;
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members
Overload List

IndexedElement(XElement,IndexingHintAction) Method
Example

The element to apply the hint to.

The action specified by the hint.

A hint to create and use an index on the specified XML element. The hint has specified action.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function IndexedElement( _`
`  ByVal element As XElement, _`
`  ByVal action As IndexingHintAction _`
`) As XHint`

C#

```
public static XHint IndexedElement(
   XElement element,
   IndexingHintAction action
)
```

Parameters

element

The element to apply the hint to.

action

The action specified by the hint.

Return Value

Formally, the hint returns the same value that it receives in the parameter. In fact, it is never executed, its role is purely declarative.

Remarks

Hints are used declaratively. They tell LiveLinq query optimizer to create and use an index on this element, if possible. After it is used for query optimization, before the query is executed, this hint is removed from the expression. See C1.LiveLinq.Hints for more details.

Example

- C#

```
var query =
    from c in customers
    where
(string)c.Element("CustomerID").IndexedElement(IndexingHintAction.Manda
tory) == "ALFKI"
    select c;
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members
Overload List

IndexedElement(XContainer,XName,IndexingHintAction) Method
Example

The element or document containing the element.

The name of the element to get.

The action specified by the hint.

A hint to create and use an index on the specified XML element. The hint has specified action.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function IndexedElement( _
   ByVal container As XContainer, _
   ByVal name As XName, _
   ByVal action As IndexingHintAction _
) As XHint
```

C#

```
public static XHint IndexedElement(
   XContainer container,
   XName name,
   IndexingHintAction action
)
```

Parameters

container

The element or document containing the element.

name

The name of the element to get.

action

The action specified by the hint.

Return Value

The element that has the specified name.

Remarks

Hints are used declaratively. They tell LiveLinq query optimizer to create and use an index on this element, if possible. When the query is executed, the hint method **IndexedElement** is replaced with the standard LINQ to XML method System.Xml.Linq.XContainer.Element(System.Xml.Linq.XName). See C1.LiveLinq.Hints for more details.

Example

- C#

```
var query =
    from c in customers
    where (string)c.IndexedElement("CustomerID",
IndexingHintAction.Mandatory) == "ALFKI"
    select c;
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members
Overload List

Nodes<T> Method
The type of the objects in the source *view*, constrained to System.Xml.Linq.XContainer.

The source view.

Returns a view representing the collection of child nodes of every document and element in the source view.

Syntax

Visual Basic (Declaration)

```
Public Shared Function Nodes(Of T As XContainer)( _
   ByVal view As View(Of T) _
) As View(Of XNode)
```

C#

```
public static View<XNode> Nodes<T>(
   View<T> view
)
where T: XContainer
```

Parameters

view

The source view.

Type Parameters

T

The type of the objects in the source *view*, constrained to System.Xml.Linq.XContainer.

Return Value

A view containing every child node of every document and element in the source view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members

Root Method
The view representing this XML document.

Gets the view for the root element of the XML tree for this document.

Syntax

Visual Basic (Declaration)

```
Public Shared Function Root( _
   ByVal documentView As View(Of XDocument) _
) As View(Of XElement)
```

C#

```
public static View<XElement> Root(
   View<XDocument> documentView
)
```

Parameters

documentView

The view representing this XML document.

Return Value

A view containing a single element, the root of the XML tree.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members

C1.LiveLinq.Metadata Namespace

Overview
Inheritance Hierarchy

See Also

Reference

C1.LiveLinq.4 Assembly

C1.WPF.LiveLinq Namespace

Overview

Classes

	Class	Description
	WpfExtensions	Provides a set of static (extension) methods for System.Collections.Specialized.INotifyCollectionChanged and ObservableCollection types (specific to WPF).

See Also

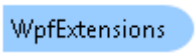
Reference

C1.LiveLinq.4 Assembly

Classes

WpfExtensions
Provides a set of static (extension) methods for System.Collections.Specialized.INotifyCollectionChanged and ObservableCollection types (specific to WPF).

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit NotInheritable Class WpfExtensions`
C#
`public static class WpfExtensions`

Inheritance Hierarchy

System.Object
C1.WPF.LiveLinq.WpfExtensions

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

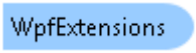
See Also

Reference

WpfExtensions Members
C1.WPF.LiveLinq Namespace

Overview
Provides a set of static (extension) methods for System.Collections.Specialized.INotifyCollectionChanged and ObservableCollection types (specific to WPF).

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit NotInheritable Class WpfExtensions`
C#
`public static class WpfExtensions`

Inheritance Hierarchy

System.Object
C1.WPF.LiveLinq.WpfExtensions

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

WpfExtensions Members
C1.WPF.LiveLinq Namespace

Members
Methods

The following tables list the members exposed by WpfExtensions.

Public Methods

	Name	Description
S	AsLive	Overloaded. Creates a view based on the specified System.Collections.Specialized.INotifyCollectionChanged data source.
S	ToIndexed	Overloaded. Creates an C1.LiveLinq.Collections.IndexedCollection<T> based on the specified System.Collections.Specialized.INotifyCollectionChanged data source.

Top

See Also

Reference

WpfExtensions Class
C1.WPF.LiveLinq Namespace

Methods
For a list of all members of this type, see WpfExtensions members.

Public Methods

	Name	Description

S	AsLive	Overloaded. Creates a view based on the specified System.Collections.Specialized.INotifyCollectionChanged data source.
S	ToIndexed	Overloaded. Creates an C1.LiveLinq.Collections.IndexedCollection<T> based on the specified System.Collections.Specialized.INotifyCollectionChanged data source.

Top

See Also

Reference

WpfExtensions Class
C1.WPF.LiveLinq Namespace

AsLive Method
Creates a view based on the specified System.Collections.Specialized.INotifyCollectionChanged data source.

Overload List

Overload	Description
AsLive<T>(INotifyCollectionChanged)	Creates a view based on the specified System.Collections.Specialized.INotifyCollectionChanged data source.
AsLive<T>(INotifyCollectionChanged,ViewOrder)	Creates a view based on the specified System.Collections.Specialized.INotifyCollectionChanged data source.
AsLive<T>(ObservableCollection<T>)	A typed specialization of the AsLive<T>(INotifyCollectionChanged) method.
AsLive<T>(ObservableCollection<T>,ViewOrder)	A typed specialization of the AsLive<T>(INotifyCollectionChanged,ViewOrder) method.

AsLive<T>(ReadOnlyObservableCollection<T>)	A typed specialization of the AsLive<T>(INotifyCollectionChanged) method.
AsLive<T>(ReadOnlyObservableCollection<T>,ViewOrder)	A typed specialization of the AsLive<T>(INotifyCollectionChanged,ViewOrder) method.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

WpfExtensions Class
WpfExtensions Members

AsLive<T>(INotifyCollectionChanged) Method
The type of the elements in the view.

The System.Collections.Specialized.INotifyCollectionChanged data source to expose as a view.

Creates a view based on the specified System.Collections.Specialized.INotifyCollectionChanged data source.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function AsLive(Of T)( _
   ByVal source As INotifyCollectionChanged _
) As View(Of T)
```

C#

```
public static View<T> AsLive<T>(
   INotifyCollectionChanged source
)
```

Parameters

source

The System.Collections.Specialized.INotifyCollectionChanged data source to expose as a view.

Type Parameters

T

The type of the elements in the view.

Return Value

A view that contains the same elements as the System.Collections.Specialized.INotifyCollectionChanged data source.

Remarks

Use this method to build views from existing data sources implementing System.Collections.Specialized.INotifyCollectionChanged. The element type of this data source must implement System.ComponentModel.INotifyPropertyChanged, see Using the built-in collection class IndexedCollection(T) (LiveLinq to Objects)|tag=Using_the_built_in_collection_class_IndexedCollectionT_LiveLinq_to_Objects .

The resulting view may have its elements ordered differently than they are ordered in the *source*. Correspondingly, views built on this resulting view (for example, if you filter it with **Where**) will not preserve the source order either. If you need to preserve the source order, consider using the other **AsLive** overload where you can specify to what extent you need the order to be preserved.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

WpfExtensions Class
WpfExtensions Members
Overload List

The type of the elements in the view.

The System.Collections.Specialized.INotifyCollectionChanged data source to expose as a view.

Specifies whether to preserve source item order.

Creates a view based on the specified System.Collections.Specialized.INotifyCollectionChanged data source.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function AsLive(Of T)( _`
`  ByVal source As INotifyCollectionChanged, _`
`  ByVal order As ViewOrder _`
`) As View(Of T)`

C#

`public static View<T> AsLive<T>(`
`  INotifyCollectionChanged source,`
`  ViewOrder order`
`)`

Parameters

source

The System.Collections.Specialized.INotifyCollectionChanged data source to expose as a view.

order

Specifies whether to preserve source item order.

Type Parameters

T

The type of the elements in the view.

Return Value

A view that contains the same elements as the System.Collections.Specialized.INotifyCollectionChanged data source.

Remarks

Use this method to build views from existing data sources implementing System.Collections.Specialized.INotifyCollectionChanged. The element type of this data source must implement System.ComponentModel.INotifyPropertyChanged, see Using the built-in collection class IndexedCollection(T) (LiveLinq to Objects)|tag=Using_the_built_in_collection_class_IndexedCollectionT_LiveLinq_to_Objects.

If the *order* parameter specifies preserving item order, the order of items in the source is preserved, at a certain performance cost, in the resulting view and in views based on it (for example, if you filter it with **Where**).

Note that **Join** does not preserve source order. If you need to order a join result, use **OrderBy** after **Join**.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

WpfExtensions Class
WpfExtensions Members
Overload List

AsLive<T>(ObservableCollection<T>) Method
The type of the elements in the view.

The ObservableCollection to expose as a view.

A typed specialization of the AsLive<T>(INotifyCollectionChanged) method.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function AsLive(Of T)( _
   ByVal source As ObservableCollection(Of T) _
) As View(Of T)
```

C#

```
public static View<T> AsLive<T>(
   ObservableCollection<T> source
)
```

Parameters

source

The ObservableCollection to expose as a view.

Type Parameters

T

The type of the elements in the view.

Return Value

A view that contains the same elements as the ObservableCollection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

WpfExtensions Class
WpfExtensions Members
Overload List

AsLive<T>(ObservableCollection<T>,ViewOrder) Method
The type of the elements in the view.

The ObservableCollection to expose as a view.

Specifies whether to preserve source item order.

A typed specialization of the AsLive<T>(INotifyCollectionChanged,ViewOrder) method.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function AsLive(Of T)( _
   ByVal source As ObservableCollection(Of T), _
   ByVal order As ViewOrder _
) As View(Of T)
```

C#

```
public static View<T> AsLive<T>(
   ObservableCollection<T> source,
   ViewOrder order
)
```

Parameters

source

The ObservableCollection to expose as a view.

order

Specifies whether to preserve source item order.

Type Parameters

T

The type of the elements in the view.

Return Value

A view that contains the same elements as the ObservableCollection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

WpfExtensions Class
WpfExtensions Members
Overload List

AsLive<T>(ReadOnlyObservableCollection<T>) Method
The type of the elements in the view.

The ReadOnlyObservableCollection to expose as a view.

A typed specialization of the AsLive<T>(INotifyCollectionChanged) method.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function AsLive(Of T)( _`
`  ByVal source As ReadOnlyObservableCollection(Of T) _`
`) As View(Of T)`

C#

`public static View<T> AsLive<T>(`
`  ReadOnlyObservableCollection<T> source`
`)`

Parameters

source

The ReadOnlyObservableCollection to expose as a view.

Type Parameters

T

The type of the elements in the view.

Return Value

A view that contains the same elements as the ReadOnlyObservableCollection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

WpfExtensions Class
WpfExtensions Members
Overload List

AsLive<T>(ReadOnlyObservableCollection<T>,ViewOrder) Method
The type of the elements in the view.

The ReadOnlyObservableCollection to expose as a view.

Specifies whether to preserve source item order.

A typed specialization of the AsLive<T>(INotifyCollectionChanged,ViewOrder) method.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function AsLive(Of T)( _`
`  ByVal source As ReadOnlyObservableCollection(Of T), _`
`  ByVal order As ViewOrder _`
`) As View(Of T)`

C#

`public static View<T> AsLive<T>(`
`  ReadOnlyObservableCollection<T> source,`
`  ViewOrder order`
`)`

Parameters

source

The ReadOnlyObservableCollection to expose as a view.

order

Specifies whether to preserve source item order.

Type Parameters

T

The type of the elements in the view.

Return Value

A view that contains the same elements as the ReadOnlyObservableCollection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

WpfExtensions Class
WpfExtensions Members
Overload List

ToIndexed Method
Creates an C1.LiveLinq.Collections.IndexedCollection<T> based on the specified System.Collections.Specialized.INotifyCollectionChanged data source.

Overload List

Overload	Description
ToIndexed<T>(INotifyCollectionChanged)	Creates an C1.LiveLinq.Collections.IndexedCollection<T> based on the specified System.Collections.Specialized.INotifyCollectionChanged data source.
ToIndexed<T>(ObservableCollection<T>)	A typed specialization of the ToIndexed<T>(INotifyCollectionChanged) method.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

WpfExtensions Class
WpfExtensions Members

ToIndexed<T>(INotifyCollectionChanged) Method
The type of the elements in the collection.

An System.Collections.Specialized.INotifyCollectionChanged data source to represent as an C1.LiveLinq.Collections.IndexedCollection<T>.

Creates an C1.LiveLinq.Collections.IndexedCollection<T> based on the specified System.Collections.Specialized.INotifyCollectionChanged data source.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function ToIndexed(Of T)( _
   ByVal source As INotifyCollectionChanged _
) As IndexedCollection(Of T)
```

C#

```
public static IndexedCollection<T> ToIndexed<T>(
   INotifyCollectionChanged source
)
```

Parameters

source

An System.Collections.Specialized.INotifyCollectionChanged data source to represent as an C1.LiveLinq.Collections.IndexedCollection<T>.

Type Parameters

T

The type of the elements in the collection.

Return Value

An C1.LiveLinq.Collections.IndexedCollection<T> that contains the same elements as the System.Collections.Specialized.INotifyCollectionChanged and enables indexing of that data source.

Remarks

Use this method to index and query your existing data sources implementing System.Collections.Specialized.INotifyCollectionChanged. The element type of this data source must implement System.ComponentModel.INotifyPropertyChanged, see Using the built-in collection class IndexedCollection(T) (LiveLinq to Objects)|tag=Using_the_built_in_collection_class_IndexedCollectionT_LiveLinq_to_Objects .

The collection returned by this method also implements the System.Collections.Specialized.INotifyCollectionChanged interface, because this method actually returns objects of an internal derived class that implements that interface.

Note: Indexes created on the resulting C1.LiveLinq.Collections.IndexedCollection<T> are owned by it and not by the original data source. Every **ToIndexed()** call creates a separate object that has its own separate indexes. Avoid calling **ToIndexed()** repeatedly for the same collection because it can increase the cost of maintaining indexes.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

WpfExtensions Class
WpfExtensions Members
Overload List

ToIndexed<T>(ObservableCollection<T>) Method
The type of the elements in the collection.

An ObservableCollection represent as an C1.LiveLinq.Collections.IndexedCollection<T>.

A typed specialization of the ToIndexed<T>(INotifyCollectionChanged) method.

Syntax

Visual Basic (Declaration)	

`Public Overloads Shared Function ToIndexed(Of T)( _`
`  ByVal source As ObservableCollection(Of T) _`
`) As IndexedCollection(Of T)`	

C#	

`public static IndexedCollection<T> ToIndexed<T>(`
`  ObservableCollection<T> source`
`)`	

Parameters

source

An ObservableCollection represent as an C1.LiveLinq.Collections.IndexedCollection<T>.

Type Parameters

T

The type of the elements in the collection.

Return Value

An C1.LiveLinq.Collections.IndexedCollection<T> that contains the same elements as the ObservableCollection and enables indexing of that collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

WpfExtensions Class
WpfExtensions Members
Overload List

C1.Silverlight.Data.Entity Assembly

Overview

%%description%%

" -->

Namespaces

Namespace	Description
C1.Data	
C1.Data.DataSource	
C1.Data.Transactions	
C1.Silverlight.Data	
C1.Silverlight.Data.RiaServices	
C1.Util.Licensing	

Namespaces

C1.Data Namespace

Overview

Classes

	Class	Description
	ClientCacheBase	Represents the client-side cache, the central hub of data access in the Studio for Entity Framework.
	ClientScope	Defines a scope of data access. Provides facilities to create client views.

	ClientView<T>	Represents a *client view*, that is, a live view that is connected to a remote source, such as a System.ServiceModel.DomainServices.Client.DomainContext.
	ClientViewLoadedEventArgs	Provides data for the ClientView<T>.Loaded event.
	DataExtensions	Extension methods provided by Studio for Entity Framework.
	FilteredView<T>	Represents a client view filtered by a filter key function, an operator and a filter key value.
	PagingView<T>	Represents a paged client view.
	ProgressiveView<T>	Represents a client view that loads entities sequentially (progressively) page by page. The user sees the result and can interact with it before all pages are loaded.
	SavedChangesEventArgs	Provides data for the C1DataSource.SavedChanges event.

See Also

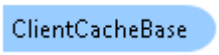
Reference

C1.Silverlight.Data.Entity Assembly

Classes

ClientCacheBase
Represents the client-side cache, the central hub of data access in the Studio for Entity Framework.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit Class ClientCacheBase`

C#

```
public abstract class ClientCacheBase
```

Remarks

Usually, a single instance of this class is created on application startup with an ObjectContext/DomainContext as a parameter and exists during the entire application lifetime, while each form, window, or user control works with data using a ClientScope created by calling the CreateScope method.

It is the base class for platform-specific implementations, such as C1.Data.Entities.EntityClientCache (Entity Framework), C1.Silverlight.Data.RiaServices.RiaClientCache (RIA Services).

Inheritance Hierarchy

System.Object
  C1.Data.ClientCacheBase
    C1.Data.Entities.EntityClientCache
    C1.Silverlight.Data.RiaServices.RiaClientCache

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

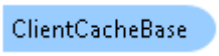
See Also

Reference

ClientCacheBase Members
C1.Data Namespace

Overview
Represents the client-side cache, the central hub of data access in the Studio for Entity Framework.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit Class ClientCacheBase`
C#
`public abstract class ClientCacheBase`

Remarks

Usually, a single instance of this class is created on application startup with an ObjectContext/DomainContext as a parameter and exists during the entire application lifetime, while each form, window, or user control works with data using a ClientScope created by calling the CreateScope method.

It is the base class for platform-specific implementations, such as C1.Data.Entities.EntityClientCache (Entity Framework), C1.Silverlight.Data.RiaServices.RiaClientCache (RIA Services).

Inheritance Hierarchy

System.Object
  C1.Data.ClientCacheBase
    C1.Data.Entities.EntityClientCache
    C1.Silverlight.Data.RiaServices.RiaClientCache

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCacheBase Members
C1.Data Namespace

Members
Methods

The following tables list the members exposed by ClientCacheBase.

Public Methods

	Name	Description
	BulkChanges	Used to group massive changes to entities and to allow manual explicit changes of entity states.
	CleanupCache	Forces unused memory to be released, unused entities to be detached from the context. It is usually done automatically, so programmers rarely need to call this method in code.
	Clear	Clears client-side cache entirely. Call this method if you want to make sure that following queries will fetch fresh data from the server.
	CreateScope	Creates a ClientScope that defines a scope of data access.
	CreateTransaction	Creates a C1.Data.Transactions.ClientTransaction that allows you to easily cancel changes made in transaction scope.
	Refresh	Refreshes data in all C1DataSource controls connected to this ClientCacheBase.
	RejectChanges	Reverts all pending changes for this ClientCacheBase. It is recommended to call this method instead of System.ServiceModel.DomainServices.Client.DomainContext.RejectChanges.
	SaveChanges	Persists all changes to the server. This is a simple shortcut calling DomainContext.SubmitChanges. If you need more detailed control over the result of the submit changes operation, you can call the System.ServiceModel.DomainServices.Client.DomainContext.SubmitChanges method directly.

Top

See Also

Reference

ClientCacheBase Class
C1.Data Namespace

Methods
For a list of all members of this type, see ClientCacheBase members.

Public Methods

	Name	Description
	BulkChanges	Used to group massive changes to entities and to allow manual explicit changes of entity states.
	CleanupCache	Forces unused memory to be released, unused entities to be detached from the context. It is usually done automatically, so programmers rarely need to call this method in code.
	Clear	Clears client-side cache entirely. Call this method if you want to make sure that following queries will fetch fresh data from the server.
	CreateScope	Creates a ClientScope that defines a scope of data access.
	CreateTransaction	Creates a C1.Data.Transactions.ClientTransaction that allows you to easily cancel changes made in transaction scope.
	Refresh	Refreshes data in all C1DataSource controls connected to this ClientCacheBase.
	RejectChanges	Reverts all pending changes for this ClientCacheBase. It is recommended to call this method instead of System.ServiceModel.DomainServices.Client.DomainContext.RejectChanges.
	SaveChanges	Persists all changes to the server. This is a simple shortcut calling DomainContext.SubmitChanges. If you need more detailed control over the result of the submit changes operation, you can call the System.ServiceModel.DomainServices.Client.DomainContext.SubmitChanges

		method directly.

Top

See Also

Reference

ClientCacheBase Class
C1.Data Namespace

BulkChanges Method
Example

A delegate that makes changes in entities.

Used to group massive changes to entities and to allow manual explicit changes of entity states.

Syntax

Visual Basic (Declaration)

`Public Sub BulkChanges( _`
`  ByVal makeChanges As Action _`
`)`

C#

`public void BulkChanges(`
`  Action makeChanges`
`)`

Parameters

makeChanges

A delegate that makes changes in entities.

Remarks

Internal state of the client-side cache and all existing client views based on the cache are kept unchanged, aren't updated while the given makeChanges is executed. After the delegate completes its execution (having modified multiple entities), the client-side cache internal state is restored and client views are updated (maintained) to reflect the changes made in entities during the delegate's execution.

There are two main scenarios where you should consider calling this method:

1. Using this method when you make a lot of changes to entities can improve performance because the change processing is deferred, occurs only once after all changes are done instead of every time on each change. Depending on the amount of changes, the speedup can be considerable.
2. You must use this method when you need to make changes to entity states by calling any of the following methods:
    - System.Data.Objects.ObjectStateEntry.ChangeState/SetModified/AcceptChanges,
    - System.Data.Objects.ObjectContext.AcceptAllChanges,
    - System.ServiceModel.DomainServices.Client.DomainContext.RejectChanges,
    - System.ServiceModel.DomainServices.Client.Entity.AcceptChanges/RejectChanges,
    - System.ServiceModel.DomainServices.Client.EntitySet.AcceptChanges/RejectChanges,
    - System.Windows.Controls.DomainDataSource.RejectChanges.

Calling these methods without wrapping them with BulkChanges can corrupt the client-side cache.

Example

- C#

```
var scope = clientCache.CreateScope();
clientCache.BulkChanges(delegate {
    foreach(var detail in scope.GetItems<Order_Details>)
        detail.Discount *= 2;
});
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCacheBase Class
ClientCacheBase Members

CleanupCache Method

Forces unused memory to be released, unused entities to be detached from the context. It is usually done automatically, so programmers rarely need to call this method in code.

Syntax

Visual Basic (Declaration)
`Public Sub CleanupCache()`
C#
`public void CleanupCache()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCacheBase Class
ClientCacheBase Members

Clear Method
Clears client-side cache entirely. Call this method if you want to make sure that following queries will fetch fresh data from the server.

Syntax

Visual Basic (Declaration)
`Public Sub Clear()`
C#
`public void Clear()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCacheBase Class
ClientCacheBase Members

CreateScope Method
Creates a ClientScope that defines a scope of data access.

Syntax

Visual Basic (Declaration)
`Public Function CreateScope() As ClientScope`
C#
`public ClientScope CreateScope()`

Return Value

A new ClientScope.

Remarks

Usually, every form, window, or user control creates a ClientScope and uses it to access entities.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCacheBase Class
ClientCacheBase Members
ClientScope Class

CreateTransaction Method
Creates a C1.Data.Transactions.ClientTransaction that allows you to easily cancel changes made in transaction scope.

Syntax

Visual Basic (Declaration)
`Public Function CreateTransaction() As ClientTransaction`
C#
`public ClientTransaction CreateTransaction()`

Return Value

A new C1.Data.Transactions.ClientTransaction

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCacheBase Class
ClientCacheBase Members

Refresh Method
Refreshes data in all C1DataSource controls connected to this ClientCacheBase.

Syntax

Visual Basic (Declaration)
`Public Sub Refresh()`
C#
`public void Refresh()`

Remarks

This method calls C1DataSource.Refresh() for every C1DataSource connected to this ClientCacheBase. Changes made on the client are preserved.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCacheBase Class
ClientCacheBase Members

RejectChanges Method
Reverts all pending changes for this ClientCacheBase. It is recommended to call this method instead of System.ServiceModel.DomainServices.Client.DomainContext.RejectChanges.

Syntax

Visual Basic (Declaration)	
`Public Sub RejectChanges()`	
C#	
`public void RejectChanges()`	

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCacheBase Class
ClientCacheBase Members

SaveChanges Method
Persists all changes to the server. This is a simple shortcut calling DomainContext.SubmitChanges. If you need more detailed control over the result of the submit changes operation, you can call the System.ServiceModel.DomainServices.Client.DomainContext.SubmitChanges method directly.

Syntax

Visual Basic (Declaration)	
`Public Sub SaveChanges()`	
C#	
`public void SaveChanges()`	

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCacheBase Class
ClientCacheBase Members

ClientScope
Defines a scope of data access. Provides facilities to create client views.

Object Model

Syntax

Visual Basic (Declaration)	
`Public Class ClientScope`	
C#	
`public class ClientScope`	

Remarks

Usually, one scope is created per form/window. Entities pinned to the scope (marked as needed) are not evicted from the cache until the scope is disposed or collected by the garbage collector.

This class is a base class for platform-specific scopes, such as C1.Data.Entities.EntityClientCache (Entity Framework) and C1.Silverlight.Data.RiaServices.RiaClientCache (RIA Services).

Inheritance Hierarchy

System.Object
  C1.Data.ClientScope
    C1.Data.Entities.EntityClientScope
    C1.Silverlight.Data.RiaServices.RiaClientScope

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientScope Members
C1.Data Namespace

Overview
Defines a scope of data access. Provides facilities to create client views.

Object Model

Syntax

Visual Basic (Declaration)
`Public Class ClientScope`
C#
`public class ClientScope`

Remarks

Usually, one scope is created per form/window. Entities pinned to the scope (marked as needed) are not evicted from the cache until the scope is disposed or collected by the garbage collector.

This class is a base class for platform-specific scopes, such as C1.Data.Entities.EntityClientCache (Entity Framework) and C1.Silverlight.Data.RiaServices.RiaClientCache (RIA Services).

Inheritance Hierarchy

System.Object
  C1.Data.ClientScope
    C1.Data.Entities.EntityClientScope
    C1.Silverlight.Data.RiaServices.RiaClientScope

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientScope Members
C1.Data Namespace

Members
Properties Methods

The following tables list the members exposed by ClientScope.

Public Constructors

	Name	Description
	ClientScope Constructor	Initializes a new instance of ClientScope class with the given ClientCacheBase.

Top

Public Properties

	Name	Description
	ClientCache	Gets the ClientCacheBase to which this client scope is connected.

Top

Public Methods

	Name	Description
	AddRef	Overloaded. Marks an entity as needed. Needed entities are not detached/released from the context until the client scope is disposed.
	Dispose	Marks the scope as disposed. Entities that were marked needed by a disposed scope may be disposed of (evicted from the cache, detached from context) unless they are needed by other active scopes.
	Release	Overloaded. Unmark a needed entity.

Top

See Also

Reference

ClientScope Class
C1.Data Namespace

ClientScope Constructor
An instance of the ClientCacheBase class to which the new client scopeis connected.

Initializes a new instance of ClientScope class with the given ClientCacheBase.

Syntax

Visual Basic (Declaration)

```
Public Function New( _
   ByVal clientCache As ClientCacheBase _
)
```

C#

```
public ClientScope(
   ClientCacheBase clientCache
```

```
)
```

Parameters

clientCache

An instance of the ClientCacheBase class to which the new client scopeis connected.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientScope Class
ClientScope Members

Methods
For a list of all members of this type, see ClientScope members.

Public Methods

	Name	Description
	AddRef	Overloaded. Marks an entity as needed. Needed entities are not detached/released from the context until the client scope is disposed.
	Dispose	Marks the scope as disposed. Entities that were marked needed by a disposed scope may be disposed of (evicted from the cache, detached from context) unless they are needed by other active scopes.
	Release	Overloaded. Unmark a needed entity.

Top

See Also

Reference

ClientScope Class
C1.Data Namespace

AddRef Method
Marks an entity as needed. Needed entities are not detached/released from the context until the client scope is disposed.

Overload List

Overload	Description
AddRef(Object)	Marks an entity as needed. Needed entities are not detached/released from the context until the client scope is disposed.
AddRef(Type)	Mark all entities of a given type as needed. All entities of that type will not be detached from the context until the client scope is disposed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientScope Class
ClientScope Members

AddRef(Object) Method
An entity to be marked as needed.

Marks an entity as needed. Needed entities are not detached/released from the context until the client scope is disposed.

Syntax

Visual Basic (Declaration)

```
Public Overloads Sub AddRef( _
   ByVal entity As Object _
```

```
)
```

C#

```
public void AddRef(
   object entity
)
```

Parameters

entity

An entity to be marked as needed.

Remarks

Client views and C1DataSource classes mark entities as needed automatically. Use this method only when you fetch entities using other means, bypassing Studio for EF classes with direct access to the underlying object context. When you no longer need those entities, call Release(Object). AddRef and Release are counting, every AddRef call must be balanced by a Release call.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientScope Class
ClientScope Members
Overload List

AddRef(Type) Method

An entity type to mark as needed.

Mark all entities of a given type as needed. All entities of that type will not be detached from the context until the client scope is disposed.

Syntax

Visual Basic (Declaration)

```
Public Overloads Sub AddRef( _
   ByVal entityType As Type _
)
```

C#

```
public void AddRef(
   Type entityType
)
```

Parameters

entityType

An entity type to mark as needed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientScope Class
ClientScope Members
Overload List

Dispose Method

Marks the scope as disposed. Entities that were marked needed by a disposed scope may be disposed of (evicted from the cache, detached from context) unless they are needed by other active scopes.

Syntax

Visual Basic (Declaration)

```
Public Sub Dispose()
```

C#

```
public void Dispose()
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientScope Class
ClientScope Members

Release Method
Unmark a needed entity.

Overload List

Overload	Description
Release(Object)	Unmark a needed entity.
Release(Type)	Unmark a needed entity type. Calling this method does not release memory.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientScope Class
ClientScope Members

Release(Object) Method
An entity that was marked as needed using AddRef(Object).

Unmark a needed entity.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function Release( _
   ByVal entity As Object _
) As Boolean
```

C#

```
public bool Release(
   object entity
)
```

Parameters

entity

An entity that was marked as needed using AddRef(Object).

Return Value

True if the entity was unmarked; otherwise, False (the entity is not unmarked until every AddRef is balanced by a Release).

Remarks

Calling this method does not release memory by itself. The entity becomes unneeded, so it can be disposed of at cache cleanup time.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientScope Class
ClientScope Members
Overload List

Release(Type) Method

An entity type that was marked as needed using AddRef(Type).

Unmark a needed entity type. Calling this method does not release memory.

Syntax

Visual Basic (Declaration)

`Public Overloads Function Release( _`
`  ByVal entityType As Type _`
`) As Boolean`

C#

`public bool Release(`
`  Type entityType`
`)`

Parameters

entityType

An entity type that was marked as needed using AddRef(Type).

Return Value

True if the entity type was unmarked (every Addref was balanced with Release); otherwise, False.

Remarks

Calling this method does not release memory until cache cleanup time.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientScope Class
ClientScope Members
Overload List

Properties
For a list of all members of this type, see ClientScope members.

Public Properties

	Name	Description
	ClientCache	Gets the ClientCacheBase to which this client scope is connected.

Top

See Also

Reference

ClientScope Class
C1.Data Namespace

ClientCache Property
Gets the ClientCacheBase to which this client scope is connected.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Property ClientCache As ClientCacheBase
```

C#

```
public ClientCacheBase ClientCache {get;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientScope Class
ClientScope Members

ClientView<T>
The type of the elements in the view.

Represents a *client view*, that is, a live view that is connected to a remote source, such as a System.ServiceModel.DomainServices.Client.DomainContext.

Object Model

Syntax

Visual Basic (Declaration)

`Public Class ClientView(Of T)`
`  Inherits C1.LiveLinq.LiveViews.View(Of T)`
`  Implements C1.LiveLinq.IObservableSource(Of T)`

C#
`public class ClientView<T> : C1.LiveLinq.LiveViews.View<T>, C1.LiveLinq.IObservableSource<T>`

Type Parameters

T

The type of the elements in the view.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.View
    C1.LiveLinq.LiveViews.View<T>
      C1.Data.ClientView<T>
        C1.Data.FilteredView<T>
        C1.Data.PagingView<T>
        C1.Data.ProgressiveView<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientView<T> Members
C1.Data Namespace

Overview
The type of the elements in the view.

Represents a *client view*, that is, a live view that is connected to a remote source, such as a System.ServiceModel.DomainServices.Client.DomainContext.

Object Model

Syntax

Visual Basic (Declaration)

`Public Class ClientView(Of T)`
`  Inherits C1.LiveLinq.LiveViews.View(Of T)`
`  Implements C1.LiveLinq.IObservableSource(Of T)`

C#
`public class ClientView<T> : C1.LiveLinq.LiveViews.View<T>, C1.LiveLinq.IObservableSource<T>`

Type Parameters

T

The type of the elements in the view.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.View

C1.LiveLinq.LiveViews.View<T>
C1.Data.ClientView<T>
C1.Data.FilteredView<T>
C1.Data.PagingView<T>
C1.Data.ProgressiveView<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientView<T> Members
C1.Data Namespace

Members
Properties Methods Events

The following tables list the members exposed by ClientView<T>.

Public Properties

	Name	Description
	AutoLoad	Gets or sets a boolean value indicating whether the client view must be loaded automatically when its data is accessed.
	Count	Gets the total number of elements in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	CurrentItem	Gets the current item in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	DeferredMaintenance	Gets the effective value of MaintenanceMode. (Inherited from C1.LiveLinq.LiveViews.View)

	IsLoaded	Gets a value indicating whether the client view is loaded.
	IsLoading	Gets a value indicating whether the client view is being loaded.
	IsReadOnly	Gets a value indicating whether this view is read-only, not updatable. (Inherited from C1.LiveLinq.LiveViews.View)
	Item	Gets the view item (element) at the specified ordinal position. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	MaintenanceMode	Gets or sets a value controlling how the view is synchronized with changes in its base data. (Inherited from C1.LiveLinq.LiveViews.View)
	MoveToFirstOnReset	Gets or sets a value indicating that the first item must be made current after initial loading or reset (on any C1.LiveLinq.SourceChangeType.Reset notification) if current item was not set by other means. The default is True. (Inherited from C1.LiveLinq.LiveViews.View)
	Order	Gets a value indicating whether and how this view preserves item order if it exists in its base data source. (Inherited from C1.LiveLinq.LiveViews.View)
	Scope	Gets the client scope to which this client view belongs.
	Transaction	Gets an instance of C1.LiveLinq.ITransaction associated with the view. If a view has a transaction associated with it, that transaction's scope is opened automatically every time the view is updated, so the programmer does not need to do it manually in code. (Inherited from C1.LiveLinq.LiveViews.View)

Top

Public Methods

	Name	Description
	AsCollectionViewFactory	Returns an instance of System.ComponentModel.ICollectionViewFactory that can be used as a source of a System.Windows.Data.CollectionViewSource. (Inherited from C1.LiveLinq.LiveViews.View)
	AsDynamic	Used for views with anonymous type constructor as the result selector, converts the C1.LiveLinq.LiveViews.View to a View<dynamic> so it can be used for data binding and programmatic access. (Inherited from C1.LiveLinq.LiveViews.View)
	AsFiltered	Filters the view on the server side using a predicate.
	AsFilteredBound<TKey>	Filters the view on the server using a key selector function and configurable value and operator.
	AttachAggregationView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	AttachView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	CancelLoad	Cancels the current loading operation.
	Concat	Overloaded. Concatenation of two views. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Contains	Determines whether the view contains a specified item. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	DeferMaintenance	Enters a defer cycle that you can use to make bulk changes to the view sources and delay automatic view maintenance. (Inherited from C1.LiveLinq.LiveViews.View)
	GetEnumerator	Returns an enumerator that iterates through the view items. (Inherited from C1.LiveLinq.LiveViews.View<T>)

	GroupBy	Overloaded. Groups the elements of a view according to a specified key selector function. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	GroupJoin	Overloaded. Correlates the elements of two views based on equality of keys and groups the results. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	IndexOf	Searches for the specified object among the elements of the view and returns the zero-based ordinal position of its first occurrence. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Join	Overloaded. Correlates the elements of two views based on matching keys. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Load	Loads the entities of the client view.
	Maintain	Brings the view up to date with its source data. (Inherited from C1.LiveLinq.LiveViews.View)
	OrderBy<TKey>	Sorts the elements of a view in ascending order. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	OrderByDescending<TKey>	Sorts the elements of a view in descending order. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Paging	Overloaded. Applies paging to this client view.
	ProgressiveLoading	Overloaded. Specifies that the client view loading is performed not in a single trip to the server but in multiple batches, each loading a page of a limited size so the user sees the result and can interact with it before all pages are loaded.
	PurgeEmptyGroups	Remove empty groups from a grouping view. (Inherited from

		C1.LiveLinq.LiveViews.View)
	Rebuild	Re-populates the view by re-executing the view's query. (Inherited from C1.LiveLinq.LiveViews.View)
	Refresh	Loads the entities of the client view ignoring the client-side cache.
	Select<TResult>	Projects each element of a view into a new form. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	SelectMany	Overloaded. Projects each element of this view to a collection of collections, flattens the resulting collections into one collection, and invokes a result selector function on each element therein. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	SetTransaction	Sets the value of the Transaction property. (Inherited from C1.LiveLinq.LiveViews.View)
	ToString	Returns a string representing this view. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Union	Overloaded. Set union of two views. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Where	Filters the source view based on a predicate. (Inherited from C1.LiveLinq.LiveViews.View<T>)

Top

Public Events

	Name	Description
	Changed	Occurs after an item of the view or the entire view has changed. (Inherited

		from C1.LiveLinq.LiveViews.View<T>)
	Loaded	Occurs when the client view has finished loading succesfully, and also when an exception has been thrown during loading.

Top

See Also

Reference

ClientView<T> Class
C1.Data Namespace

Methods
For a list of all members of this type, see ClientView<T> members.

Public Methods

	Name	Description
	AsCollectionViewFactory	Returns an instance of System.ComponentModel.ICollectionViewFactory that can be used as a source of a System.Windows.Data.CollectionViewSource. (Inherited from C1.LiveLinq.LiveViews.View)
	AsDynamic	Used for views with anonymous type constructor as the result selector, converts the C1.LiveLinq.LiveViews.View to a View<dynamic> so it can be used for data binding and programmatic access. (Inherited from C1.LiveLinq.LiveViews.View)
	AsFiltered	Filters the view on the server side using a predicate.
	AsFilteredBound<TKey>	Filters the view on the server using a key selector function and configurable value and operator.
	AttachAggregationView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<T>)

	AttachView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	CancelLoad	Cancels the current loading operation.
	Concat	Overloaded. Concatenation of two views. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Contains	Determines whether the view contains a specified item. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	DeferMaintenance	Enters a defer cycle that you can use to make bulk changes to the view sources and delay automatic view maintenance. (Inherited from C1.LiveLinq.LiveViews.View)
	GetEnumerator	Returns an enumerator that iterates through the view items. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	GroupBy	Overloaded. Groups the elements of a view according to a specified key selector function. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	GroupJoin	Overloaded. Correlates the elements of two views based on equality of keys and groups the results. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	IndexOf	Searches for the specified object among the elements of the view and returns the zero-based ordinal position of its first occurrence. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Join	Overloaded. Correlates the elements of two views based on matching keys. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Load	Loads the entities of the client view.
	Maintain	Brings the view up to date with its source data. (Inherited from

		C1.LiveLinq.LiveViews.View)
	OrderBy<TKey>	Sorts the elements of a view in ascending order. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	OrderByDescending<TKey>	Sorts the elements of a view in descending order. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Paging	Overloaded. Applies paging to this client view.
	ProgressiveLoading	Overloaded. Specifies that the client view loading is performed not in a single trip to the server but in multiple batches, each loading a page of a limited size so the user sees the result and can interact with it before all pages are loaded.
	PurgeEmptyGroups	Remove empty groups from a grouping view. (Inherited from C1.LiveLinq.LiveViews.View)
	Rebuild	Re-populates the view by re-executing the view's query. (Inherited from C1.LiveLinq.LiveViews.View)
	Refresh	Loads the entities of the client view ignoring the client-side cache.
	Select<TResult>	Projects each element of a view into a new form. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	SelectMany	Overloaded. Projects each element of this view to a collection of collections, flattens the resulting collections into one collection, and invokes a result selector function on each element therein. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	SetTransaction	Sets the value of the Transaction property. (Inherited from C1.LiveLinq.LiveViews.View)

	ToString	Returns a string representing this view. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Union	Overloaded. Set union of two views. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Where	Filters the source view based on a predicate. (Inherited from C1.LiveLinq.LiveViews.View<T>)

Top

See Also

Reference

ClientView<T> Class
C1.Data Namespace

AsFiltered Method
A function to apply each element to test the condition.

Filters the view on the server side using a predicate.

Syntax

Visual Basic (Declaration)

```
Public Overridable Function AsFiltered( _
   ByVal predicate As Expression(Of Func(Of T,Boolean)) _
) As ClientView(Of T)
```

C#

```
public virtual ClientView<T> AsFiltered(
   Expression<Func<T,bool>> predicate
)
```

Parameters

predicate

A function to apply each element to test the condition.

Return Value

A client view that contains elements of this view that satisfy the predicate.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientView<T> Class
ClientView<T> Members

AsFilteredBound<TKey> Method
The type of the value used for filtering.

A function that returns a key value for filtering given a view item.

Filters the view on the server using a key selector function and configurable value and operator.

Syntax

Visual Basic (Declaration)

`Public Overridable Function AsFilteredBound(Of TKey)( _`
`  ByVal keySelector As Expression(Of Func(Of T,TKey)) _`
`) As FilteredView(Of T)`

C#

`public virtual FilteredView<T> AsFilteredBound<TKey>(`
`  Expression<Func<T,TKey>> keySelector`
`)`

Parameters

keySelector

A function that returns a key value for filtering given a view item.

Type Parameters

TKey

The type of the value used for filtering.

Return Value

A FilteredView<T> that contains elements of this view that have keys satisfying the condition.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientView<T> Class
ClientView<T> Members

CancelLoad Method

Cancels the current loading operation.

Syntax

Visual Basic (Declaration)
`Public Sub CancelLoad()`
C#
`public void CancelLoad()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientView<T> Class
ClientView<T> Members

Load Method
Loads the entities of the client view.

Syntax

Visual Basic (Declaration)
`Public Sub Load()`
C#
`public void Load()`

Remarks

If the entities are already in the cache, there will be no roundtrip to the server.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientView<T> Class
ClientView<T> Members

Paging Method
Applies paging to this client view.

Overload List

Overload	Description
Paging<TKey>(Expression<Func<T,TKey>>,Int32)	Applies paging to this client view.
Paging<TKey>(Expression<Func<T,TKey>>,Boolean,Int32)	Applies paging to this client view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientView<T> Class
ClientView<T> Members

Paging<TKey>(Expression<Func<T,TKey>>,Int32) Method
The type of the sort key.

A function specifying a sort key.

A value for the PageSize property, the number of items to load in a page.

Applies paging to this client view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function Paging(Of TKey)( _
   ByVal sortKeySelector As Expression(Of Func(Of T,TKey)), _
   ByVal pageSize As Integer _
) As PagingView(Of T)
```

C#

```
public PagingView<T> Paging<TKey>(
   Expression<Func<T,TKey>> sortKeySelector,
   int pageSize
)
```

Parameters

sortKeySelector

A function specifying a sort key.

pageSize

A value for the PageSize property, the number of items to load in a page.

Type Parameters

TKey

The type of the sort key.

Return Value

A paged client view.

Remarks

Sorting is required, paging entities is impossible without sort.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientView<T> Class
ClientView<T> Members
Overload List

Paging<TKey>(Expression<Func<T,TKey>>,Boolean,Int32) Method
The type of the sort key.

A function specifying a sort key.

A boolean value indicating whether sorting must be performed in the ascending order (descending, if false).

A value for the PageSize property, the number of items to load in a page.

Applies paging to this client view.

Syntax

Visual Basic (Declaration)	

```
Public Overloads Function Paging(Of TKey)( _
   ByVal sortKeySelector As Expression(Of Func(Of T,TKey)), _
   ByVal ascending As Boolean, _
   ByVal pageSize As Integer _
) As PagingView(Of T)
```

C#

```
public PagingView<T> Paging<TKey>(
   Expression<Func<T,TKey>> sortKeySelector,
   bool ascending,
   int pageSize
)
```

Parameters

sortKeySelector

A function specifying a sort key.

ascending

A boolean value indicating whether sorting must be performed in the ascending order (descending, if false).

pageSize

A value for the PageSize property, the number of items to load in a page.

Type Parameters

TKey

The type of the sort key.

Return Value

A paged client view.

Remarks

Sorting is required, paging entities is impossible without sort.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientView<T> Class
ClientView<T> Members
Overload List

ProgressiveLoading Method
Specifies that the client view loading is performed not in a single trip to the server but in multiple batches, each loading a page of a limited size so the user sees the result and can interact with it before all pages are loaded.

Overload List

Overload	Description
ProgressiveLoading<TKey>(Expression<Func<T,TKey>>,Int32)	Specifies that the client view loading is performed not in a single trip to the server but in multiple batches, each loading a page of a limited size so the user sees the result and can interact with it before all pages are loaded.
ProgressiveLoading<TKey>(Expression<Func<T,TKey>>,Boolean,Int32)	Specifies that the client view loading is performed not in a single trip to the server but in multiple batches, each loading a page of a limited size so the user sees the result and can interact with it before

	all pages are loaded.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientView<T> Class
ClientView<T> Members

ProgressiveLoading<TKey>(Expression<Func<T,TKey>>,Int32) Method
The type of the sort key.

A function specifying a sort key.

The size of the page.

Specifies that the client view loading is performed not in a single trip to the server but in multiple batches, each loading a page of a limited size so the user sees the result and can interact with it before all pages are loaded.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function ProgressiveLoading(Of TKey)( _
   ByVal sortKeySelector As Expression(Of Func(Of T,TKey)), _
   ByVal loadSize As Integer _
) As ProgressiveView(Of T)
```

C#

```
public ProgressiveView<T> ProgressiveLoading<TKey>(
   Expression<Func<T,TKey>> sortKeySelector,
   int loadSize
)
```

Parameters

sortKeySelector

A function specifying a sort key.

loadSize

The size of the page.

Type Parameters

TKey

The type of the sort key.

Return Value

A client view that loads the same entities as the source view but does it progressively.

Remarks

Sorting is required, loading entities progressively is impossible without sort.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientView<T> Class
ClientView<T> Members
Overload List

ProgressiveLoading<TKey>(Expression<Func<T,TKey>>,Boolean,Int32) Method
The type of the sort key.

A function specifying a sort key.

A boolean value indicating whether sorting must be performed in the ascending order (descending, if false).

The size of the page.

Specifies that the client view loading is performed not in a single trip to the server but in multiple batches, each loading a page of a limited size so the user sees the result and can interact with it before all pages are loaded.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function ProgressiveLoading(Of TKey)( _
   ByVal sortKeySelector As Expression(Of Func(Of T,TKey)), _
   ByVal ascending As Boolean, _
   ByVal loadSize As Integer _
) As ProgressiveView(Of T)
```

C#

```
public ProgressiveView<T> ProgressiveLoading<TKey>(
   Expression<Func<T,TKey>> sortKeySelector,
   bool ascending,
   int loadSize
)
```

Parameters

sortKeySelector

A function specifying a sort key.

ascending

A boolean value indicating whether sorting must be performed in the ascending order (descending, if false).

loadSize

The size of the page.

Type Parameters

TKey

The type of the sort key.

Return Value

A client view that loads the same entities as the source view but does it progressively.

Remarks

Sorting is required, loading entities progressively is impossible without sort.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientView<T> Class
ClientView<T> Members
Overload List

Refresh Method

Loads the entities of the client view ignoring the client-side cache.

Syntax

Visual Basic (Declaration)
`Public Sub Refresh()`
C#
`public void Refresh()`

Remarks

Use this method to refresh data with any changes that may have occurred on the server

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientView<T> Class
ClientView<T> Members

Properties
For a list of all members of this type, see ClientView<T> members.

Public Properties

	Name	Description
	AutoLoad	Gets or sets a boolean value indicating whether the client view must be loaded automatically when its data is accessed.
	Count	Gets the total number of elements in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	CurrentItem	Gets the current item in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	DeferredMaintenance	Gets the effective value of MaintenanceMode. (Inherited from C1.LiveLinq.LiveViews.View)
	IsLoaded	Gets a value indicating whether the client view is loaded.
	IsLoading	Gets a value indicating whether the client view is being loaded.
	IsReadOnly	Gets a value indicating whether this view is read-only, not updatable. (Inherited from C1.LiveLinq.LiveViews.View)
	Item	Gets the view item (element) at the specified ordinal position. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	MaintenanceMode	Gets or sets a value controlling how the view is synchronized with

		changes in its base data. (Inherited from C1.LiveLinq.LiveViews.View)
	MoveToFirstOnReset	Gets or sets a value indicating that the first item must be made current after initial loading or reset (on any C1.LiveLinq.SourceChangeType.Reset notification) if current item was not set by other means. The default is True. (Inherited from C1.LiveLinq.LiveViews.View)
	Order	Gets a value indicating whether and how this view preserves item order if it exists in its base data source. (Inherited from C1.LiveLinq.LiveViews.View)
	Scope	Gets the client scope to which this client view belongs.
	Transaction	Gets an instance of C1.LiveLinq.ITransaction associated with the view. If a view has a transaction associated with it, that transaction's scope is opened automatically every time the view is updated, so the programmer does not need to do it manually in code. (Inherited from C1.LiveLinq.LiveViews.View)

Top

See Also

Reference

ClientView<T> Class
C1.Data Namespace

AutoLoad Property
Gets or sets a boolean value indicating whether the client view must be loaded automatically when its data is accessed.

Syntax

Visual Basic (Declaration)
`Public Property AutoLoad As Boolean`

C#

```
public bool AutoLoad {get; set;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientView<T> Class
ClientView<T> Members

IsLoaded Property
Gets a value indicating whether the client view is loaded.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Property IsLoaded As Boolean
```

C#

```
public bool IsLoaded {get;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientView<T> Class
ClientView<T> Members

IsLoading Property
Gets a value indicating whether the client view is being loaded.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Property IsLoading As Boolean
```

C#

```
public bool IsLoading {get;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientView<T> Class
ClientView<T> Members

Scope Property
Gets the client scope to which this client view belongs.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Property Scope As ClientScope
```

C#

```
public ClientScope Scope {get;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientView<T> Class
ClientView<T> Members

Events
>

Name	Description
Changed	Occurs after an item of the view or the entire view has changed. (Inherited from C1.LiveLinq.LiveViews.View<T>)
Loaded	Occurs when the client view has finished loading succesfully, and also when an exception has been thrown during loading.

Top

See Also

Reference

ClientView<T> Class
C1.Data Namespace

Loaded Event
Occurs when the client view has finished loading succesfully, and also when an exception has been thrown during loading.

Syntax

Visual Basic (Declaration)
`Public Event Loaded As EventHandler(Of ClientViewLoadedEventArgs)`
C#
`public event EventHandler<ClientViewLoadedEventArgs> Loaded`

Event Data

The event handler receives an argument of type ClientViewLoadedEventArgs containing data related to this event. The following **ClientViewLoadedEventArgs** properties provide information specific to this event.

Property	Description
Error	Gets the loading error if the loading failed.
HasError	Gets a value indicating whether the loading has failed. If true, inspect the Error property for details.
IsErrorHandled	Gets a value indicating whether the loading error has been marked as handled by calling MarkErrorAsHandled.
Items	Gets all entities loaded by a client view.
TotalItemCount	Gets the total number of rows for the original query without any paging applied to it.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

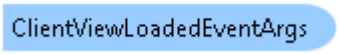
See Also

Reference

ClientView<T> Class
ClientView<T> Members

ClientViewLoadedEventArgs
Provides data for the ClientView<T>.Loaded event.

Object Model



Syntax

Visual Basic (Declaration)	

```
Public Class ClientViewLoadedEventArgs
   Inherits System.EventArgs
```

C#

```
public class ClientViewLoadedEventArgs : System.EventArgs
```

Inheritance Hierarchy

System.Object
  System.EventArgs
    C1.Data.ClientViewLoadedEventArgs

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

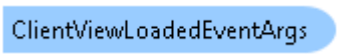
See Also

Reference

ClientViewLoadedEventArgs Members
C1.Data Namespace

Overview
Provides data for the ClientView<T>.Loaded event.

Object Model



Syntax

Visual Basic (Declaration)

```
Public Class ClientViewLoadedEventArgs
   Inherits System.EventArgs
```

C#

```
public class ClientViewLoadedEventArgs : System.EventArgs
```

Inheritance Hierarchy

System.Object
  System.EventArgs
    C1.Data.ClientViewLoadedEventArgs

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewLoadedEventArgs Members
C1.Data Namespace

Members
Properties Methods

The following tables list the members exposed by ClientViewLoadedEventArgs.

Public Properties

	Name	Description
	Error	Gets the loading error if the loading failed.
	HasError	Gets a value indicating whether the loading has failed. If true, inspect the Error property for details.
	IsErrorHandled	Gets a value indicating whether the loading error has been marked as handled by calling MarkErrorAsHandled.
	Items	Gets all entities loaded by a client view.
	TotalItemCount	Gets the total number of rows for the original query without any paging applied to it.

	ValidationErrors	Gets the validation errors.

Top

Public Methods

	Name	Description
	MarkErrorAsHandled	For the case where HasError is true, this method marks the error as handled. If this method is not called, an exception will be thrown.

Top

See Also

Reference

ClientViewLoadedEventArgs Class
C1.Data Namespace

Methods
For a list of all members of this type, see ClientViewLoadedEventArgs members.

Public Methods

	Name	Description
	MarkErrorAsHandled	For the case where HasError is true, this method marks the error as handled. If this method is not called, an exception will be thrown.

Top

See Also

Reference

ClientViewLoadedEventArgs Class
C1.Data Namespace

MarkErrorAsHandled Method
For the case where HasError is true, this method marks the error as handled. If this method is not called, an exception will be thrown.

Syntax

Visual Basic (Declaration)
`Public Sub MarkErrorAsHandled()`
C#
`public void MarkErrorAsHandled()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewLoadedEventArgs Class
ClientViewLoadedEventArgs Members

Properties
For a list of all members of this type, see ClientViewLoadedEventArgs members.

Public Properties

	Name	Description
	Error	Gets the loading error if the loading failed.
	HasError	Gets a value indicating whether the loading has failed. If true, inspect the Error property for details.
	IsErrorHandled	Gets a value indicating whether the loading error has been marked as handled by calling MarkErrorAsHandled.
	Items	Gets all entities loaded by a client view.

	TotalItemCount	Gets the total number of rows for the original query without any paging applied to it.
	ValidationErrors	Gets the validation errors.

Top

See Also

Reference

ClientViewLoadedEventArgs Class
C1.Data Namespace

Error Property
Gets the loading error if the loading failed.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Error As Exception`
C#
`public Exception Error {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewLoadedEventArgs Class
ClientViewLoadedEventArgs Members

HasError Property
Gets a value indicating whether the loading has failed. If true, inspect the Error property for details.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property HasError As Boolean`
C#
`public bool HasError {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewLoadedEventArgs Class
ClientViewLoadedEventArgs Members

IsErrorHandled Property
Gets a value indicating whether the loading error has been marked as handled by calling MarkErrorAsHandled.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property IsErrorHandled As Boolean`
C#
`public bool IsErrorHandled {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewLoadedEventArgs Class
ClientViewLoadedEventArgs Members

Items Property
Gets all entities loaded by a client view.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Items As IEnumerable`
C#
`public IEnumerable Items {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewLoadedEventArgs Class
ClientViewLoadedEventArgs Members

TotalItemCount Property
Gets the total number of rows for the original query without any paging applied to it.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property TotalItemCount As Integer`
C#
`public int TotalItemCount {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewLoadedEventArgs Class
ClientViewLoadedEventArgs Members

ValidationErrors Property
Gets the validation errors.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property ValidationErrors As IEnumerable(Of ValidationResult)`
C#
`public IEnumerable<ValidationResult> ValidationErrors {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

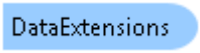
See Also

Reference

ClientViewLoadedEventArgs Class
ClientViewLoadedEventArgs Members

DataExtensions
Extension methods provided by Studio for Entity Framework.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit NotInheritable Class DataExtensions`
C#
`public static class DataExtensions`

Inheritance Hierarchy

System.Object
  C1.Data.DataExtensions

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

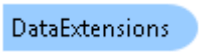
See Also

Reference

DataExtensions Members
C1.Data Namespace

Overview
Extension methods provided by Studio for Entity Framework.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit NotInheritable Class DataExtensions`
C#

```
public static class DataExtensions
```

Inheritance Hierarchy

System.Object
  C1.Data.DataExtensions

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

DataExtensions Members
C1.Data Namespace

Members
Methods

The following tables list the members exposed by DataExtensions.

Public Methods

	Name	Description
S	ExecuteIn	Overloaded. Executes a **query** in a **client scope**, so the loaded entities are pinned to the **scope** (marked as needed) by calling AddRef(Object) for each of them.

Top

See Also

Reference

DataExtensions Class
C1.Data Namespace

Methods
For a list of all members of this type, see DataExtensions members.

Public Methods

	Name	Description
S	ExecuteIn	Overloaded. Executes a **query** in a **client scope**, so the loaded entities are pinned to the **scope** (marked as needed) by calling AddRef(Object) for each of them.

Top

See Also

Reference

DataExtensions Class
C1.Data Namespace

ExecuteIn Method
Executes a **query** in a **client scope**, so the loaded entities are pinned to the **scope** (marked as needed) by calling AddRef(Object) for each of them.

Overload List

Overload	Description
ExecuteIn<T>(IEnumerable<T>,ClientScope)	Executes a query in a scope, so the loaded entities are pinned to the scope (marked as needed) by calling AddRef(Object) for each of them.
ExecuteIn<T>(EntityQuery<T>,RiaClientScope)	Executes a query in a scope, so the loaded entities are pinned to the scope (marked as needed) by calling AddRef(Object) for each of them.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

DataExtensions Class
DataExtensions Members

ExecuteIn<T>(IEnumerable<T>,ClientScope) Method
The type of items returned by the query.

The query to execute inside the client scope.

The client scope to execute the query in.

Executes a query in a scope, so the loaded entities are pinned to the scope (marked as needed) by calling AddRef(Object) for each of them.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function ExecuteIn(Of T)( _`
`  ByVal query As IEnumerable(Of T), _`
`  ByVal scope As ClientScope _`
`) As IEnumerable(Of T)`

C#

`public static IEnumerable<T> ExecuteIn<T>(`
`  IEnumerable<T> query,`
`  ClientScope scope`
`)`

Parameters

query

The query to execute inside the client scope.

scope

The client scope to execute the query in.

Type Parameters

T

The type of items returned by the query.

Return Value

The query that will be executed inside the given scope.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

DataExtensions Class
DataExtensions Members
Overload List

ExecuteIn<T>(EntityQuery<T>,RiaClientScope) Method
The type of the item in the query.

The query to execute inside the scope.

The client scope to execute the query in.

Executes a query in a scope, so the loaded entities are pinned to the scope (marked as needed) by calling AddRef(Object) for each of them.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function ExecuteIn(Of T As Entity)( _
   ByVal query As EntityQuery(Of T), _
   ByVal scope As RiaClientScope _
) As LoadOperation(Of T)
```

C#

```
public static LoadOperation<T> ExecuteIn<T>(
   EntityQuery<T> query,
   RiaClientScope scope
)
where T: Entity
```

Parameters

query

The query to execute inside the scope.

scope

The client scope to execute the query in.

Type Parameters

T

The type of the item in the query.

Return Value

The LoadOperation object representing the execution of the query.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

DataExtensions Class
DataExtensions Members
Overload List

FilteredView<T>
The type of the entities in this client view.

Represents a client view filtered by a filter key function, an operator and a filter key value.

Object Model

Syntax

Visual Basic (Declaration)	

`Public Class FilteredView(Of T)`
`  Inherits ClientView(Of T)`
`  Implements C1.LiveLinq.IObservableSource(Of T)`	

C#	
`public class FilteredView<T> : ClientView<T>, C1.LiveLinq.IObservableSource<T>`	

Type Parameters

T

The type of the entities in this client view.

Remarks

Filtering is performed on the server.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.View
    C1.LiveLinq.LiveViews.View<T>
      C1.Data.ClientView<T>
        C1.Data.FilteredView<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

FilteredView<T> Members
C1.Data Namespace

Overview
The type of the entities in this client view.

Represents a client view filtered by a filter key function, an operator and a filter key value.

Object Model

Syntax

Visual Basic (Declaration)

`Public Class FilteredView(Of T)`
`  Inherits ClientView(Of T)`
`  Implements C1.LiveLinq.IObservableSource(Of T)`

C#
`public class FilteredView<T> : ClientView<T>, C1.LiveLinq.IObservableSource<T>`

Type Parameters

T

The type of the entities in this client view.

Remarks

Filtering is performed on the server.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.View
    C1.LiveLinq.LiveViews.View<T>
      C1.Data.ClientView<T>
        C1.Data.FilteredView<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

FilteredView<T> Members
C1.Data Namespace

Members
Fields Properties Methods Events

The following tables list the members exposed by FilteredView<T>.

Public Fields

	Name	Description
S	Unfiltered	A special value indicating that filtering must not be performed. To disable filtering, assign the value of this field to the FilterKey property.

Top

Public Properties

	Name	Description
	AutoLoad	Gets or sets a boolean value indicating whether the client view must be loaded automatically when its data is accessed. (Inherited from C1.Data.ClientView<T>)
	Count	Gets the total number of elements in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	CurrentItem	Gets the current item in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	DeferredMaintenance	Gets the effective value of MaintenanceMode. (Inherited from C1.LiveLinq.LiveViews.View)
	FilterKey	Gets or sets a value that is used to filter items by comparing this value to the result of the filter key function applied to an item.

	FilterKeyType	Gets the filter key type. It is determined by the filter key function.
	IsLoaded	Gets a value indicating whether the client view is loaded. (Inherited from C1.Data.ClientView<T>)
	IsLoading	Gets a value indicating whether the client view is being loaded. (Inherited from C1.Data.ClientView<T>)
	IsReadOnly	Gets a value indicating whether this view is read-only, not updatable. (Inherited from C1.LiveLinq.LiveViews.View)
	Item	Gets the view item (element) at the specified ordinal position. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	MaintenanceMode	Gets or sets a value controlling how the view is synchronized with changes in its base data. (Inherited from C1.LiveLinq.LiveViews.View)
	MoveToFirstOnReset	Gets or sets a value indicating that the first item must be made current after initial loading or reset (on any C1.LiveLinq.SourceChangeType.Reset notification) if current item was not set by other means. The default is True. (Inherited from C1.LiveLinq.LiveViews.View)
	Operator	Gets a System.Windows.Controls.FilterOperator used to compare filter keys.
	Order	Gets a value indicating whether and how this view preserves item order if it exists in its base data source. (Inherited from C1.LiveLinq.LiveViews.View)
	Scope	Gets the client scope to which this client view belongs. (Inherited from C1.Data.ClientView<T>)
	Transaction	Gets an instance of C1.LiveLinq.ITransaction associated with the view. If a view has a transaction associated with it, that transaction's scope is

		opened automatically every time the view is updated, so the programmer does not need to do it manually in code. (Inherited from C1.LiveLinq.LiveViews.View)

Top

Public Methods

	Name	Description
	AsCollectionViewFactory	Returns an instance of System.ComponentModel.ICollectionViewFactory that can be used as a source of a System.Windows.Data.CollectionViewSource. (Inherited from C1.LiveLinq.LiveViews.View)
	AsDynamic	Used for views with anonymous type constructor as the result selector, converts the C1.LiveLinq.LiveViews.View to a View<dynamic> so it can be used for data binding and programmatic access. (Inherited from C1.LiveLinq.LiveViews.View)
	AsFiltered	Filters the view on the server side using a predicate. (Inherited from C1.Data.ClientView<T>)
	AsFilteredBound<TKey>	Filters the view on the server using a key selector function and configurable value and operator. (Inherited from C1.Data.ClientView<T>)
	AttachAggregationView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	AttachView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	BindFilterKey	Overloaded. Binds the FilterKey property using the specified System.Windows.Data.Binding object.
	CancelLoad	Cancels the current loading operation. (Inherited from

		C1.Data.ClientView<T>)
	Concat	Overloaded. Concatenation of two views. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Contains	Determines whether the view contains a specified item. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	DeferMaintenance	Enters a defer cycle that you can use to make bulk changes to the view sources and delay automatic view maintenance. (Inherited from C1.LiveLinq.LiveViews.View)
	GetEnumerator	Returns an enumerator that iterates through the view items. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	GroupBy	Overloaded. Groups the elements of a view according to a specified key selector function. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	GroupJoin	Overloaded. Correlates the elements of two views based on equality of keys and groups the results. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	IndexOf	Searches for the specified object among the elements of the view and returns the zero-based ordinal position of its first occurrence. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Join	Overloaded. Correlates the elements of two views based on matching keys. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Load	Loads the entities of the client view. (Inherited from C1.Data.ClientView<T>)
	Maintain	Brings the view up to date with its source data. (Inherited from

		C1.LiveLinq.LiveViews.View)
	OrderBy<TKey>	Sorts the elements of a view in ascending order. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	OrderByDescending<TKey>	Sorts the elements of a view in descending order. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Paging	Overloaded. Applies paging to this client view. (Inherited from C1.Data.ClientView<T>)
	ProgressiveLoading	Overloaded. Specifies that the client view loading is performed not in a single trip to the server but in multiple batches, each loading a page of a limited size so the user sees the result and can interact with it before all pages are loaded. (Inherited from C1.Data.ClientView<T>)
	PurgeEmptyGroups	Remove empty groups from a grouping view. (Inherited from C1.LiveLinq.LiveViews.View)
	Rebuild	Re-populates the view by re-executing the view's query. (Inherited from C1.LiveLinq.LiveViews.View)
	Refresh	Loads the entities of the client view ignoring the client-side cache. (Inherited from C1.Data.ClientView<T>)
	Select<TResult>	Projects each element of a view into a new form. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	SelectMany	Overloaded. Projects each element of this view to a collection of collections, flattens the resulting collections into one collection, and invokes a result selector function on each element therein. (Inherited from C1.LiveLinq.LiveViews.View<T>)

	SetTransaction	Sets the value of the Transaction property. (Inherited from C1.LiveLinq.LiveViews.View)
	ToString	Returns a string representing this view. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Union	Overloaded. Set union of two views. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Where	Filters the source view based on a predicate. (Inherited from C1.LiveLinq.LiveViews.View<T>)

Top

Public Events

	Name	Description
	Changed	Occurs after an item of the view or the entire view has changed. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Loaded	Occurs when the client view has finished loading succesfully, and also when an exception has been thrown during loading. (Inherited from C1.Data.ClientView<T>)

Top

See Also

Reference

FilteredView<T> Class
C1.Data Namespace

Methods
For a list of all members of this type, see FilteredView<T> members.

Public Methods

	Name	Description
	AsCollectionViewFactory	Returns an instance of System.ComponentModel.ICollectionViewFactory that can be used as a source of a System.Windows.Data.CollectionViewSource. (Inherited from C1.LiveLinq.LiveViews.View)
	AsDynamic	Used for views with anonymous type constructor as the result selector, converts the C1.LiveLinq.LiveViews.View to a View<dynamic> so it can be used for data binding and programmatic access. (Inherited from C1.LiveLinq.LiveViews.View)
	AsFiltered	Filters the view on the server side using a predicate. (Inherited from C1.Data.ClientView<T>)
	AsFilteredBound<TKey>	Filters the view on the server using a key selector function and configurable value and operator. (Inherited from C1.Data.ClientView<T>)
	AttachAggregationView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	AttachView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	BindFilterKey	Overloaded. Binds the FilterKey property using the specified System.Windows.Data.Binding object.
	CancelLoad	Cancels the current loading operation. (Inherited from C1.Data.ClientView<T>)
	Concat	Overloaded. Concatenation of two views. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Contains	Determines whether the view contains a specified item. (Inherited from C1.LiveLinq.LiveViews.View<T>)

	DeferMaintenance	Enters a defer cycle that you can use to make bulk changes to the view sources and delay automatic view maintenance. (Inherited from C1.LiveLinq.LiveViews.View)
	GetEnumerator	Returns an enumerator that iterates through the view items. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	GroupBy	Overloaded. Groups the elements of a view according to a specified key selector function. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	GroupJoin	Overloaded. Correlates the elements of two views based on equality of keys and groups the results. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	IndexOf	Searches for the specified object among the elements of the view and returns the zero-based ordinal position of its first occurrence. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Join	Overloaded. Correlates the elements of two views based on matching keys. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Load	Loads the entities of the client view. (Inherited from C1.Data.ClientView<T>)
	Maintain	Brings the view up to date with its source data. (Inherited from C1.LiveLinq.LiveViews.View)
	OrderBy<TKey>	Sorts the elements of a view in ascending order. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	OrderByDescending<TKey>	Sorts the elements of a view in descending order. (Inherited from C1.LiveLinq.LiveViews.View<T>)

	Paging	Overloaded. Applies paging to this client view. (Inherited from C1.Data.ClientView<T>)
	ProgressiveLoading	Overloaded. Specifies that the client view loading is performed not in a single trip to the server but in multiple batches, each loading a page of a limited size so the user sees the result and can interact with it before all pages are loaded. (Inherited from C1.Data.ClientView<T>)
	PurgeEmptyGroups	Remove empty groups from a grouping view. (Inherited from C1.LiveLinq.LiveViews.View)
	Rebuild	Re-populates the view by re-executing the view's query. (Inherited from C1.LiveLinq.LiveViews.View)
	Refresh	Loads the entities of the client view ignoring the client-side cache. (Inherited from C1.Data.ClientView<T>)
	Select<TResult>	Projects each element of a view into a new form. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	SelectMany	Overloaded. Projects each element of this view to a collection of collections, flattens the resulting collections into one collection, and invokes a result selector function on each element therein. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	SetTransaction	Sets the value of the Transaction property. (Inherited from C1.LiveLinq.LiveViews.View)
	ToString	Returns a string representing this view. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Union	Overloaded. Set union of two views. (Inherited from C1.LiveLinq.LiveViews.View<T>)

	Where	Filters the source view based on a predicate. (Inherited from C1.LiveLinq.LiveViews.View<T>)

Top

See Also

Reference

FilteredView<T> Class
C1.Data Namespace

BindFilterKey Method
Binds the FilterKey property using the specified System.Windows.Data.Binding object.

Overload List

Overload	Description
BindFilterKey(Binding)	Binds the FilterKey property using the specified System.Windows.Data.Binding object.
BindFilterKey(Object,String)	Binds the FilterKey property to a given path on a given source.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

FilteredView<T> Class
FilteredView<T> Members

BindFilterKey(Binding) Method
The System.Windows.Data.Binding object used to bind the FilterKey. Use null to unbind the previously bound FilterKey.

Binds the FilterKey property using the specified System.Windows.Data.Binding object.

Syntax

Visual Basic (Declaration)

`Public Overloads Function BindFilterKey( _`
`  ByVal binding As Binding _`
`) As FilteredView(Of T)`

C#

`public FilteredView<T> BindFilterKey(`
`  Binding binding`
`)`

Parameters

binding

The System.Windows.Data.Binding object used to bind the FilterKey. Use null to unbind the previously bound FilterKey.

Return Value

The same FilteredView<T> on which this method was called.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

FilteredView<T> Class
FilteredView<T> Members
Overload List

BindFilterKey(Object,String) Method
The object to bind to. Cannot be null.

The property path to bind to.

Binds the FilterKey property to a given path on a given source.

Syntax

Visual Basic (Declaration)

`Public Overloads Function BindFilterKey( _`
`  ByVal source As Object, _`
`  ByVal path As String _`
`) As FilteredView(Of T)`

C#

`public FilteredView<T> BindFilterKey(`
`  object source,`
`  string path`
`)`

Parameters

source

The object to bind to. Cannot be null.

path

The property path to bind to.

Return Value

The same FilteredView<T> on which this method was called.

Exceptions

Exception	Description
System.NullReferenceException	source is null.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

FilteredView<T> Class
FilteredView<T> Members
Overload List

Properties
For a list of all members of this type, see FilteredView<T> members.

Public Properties

	Name	Description
	AutoLoad	Gets or sets a boolean value indicating whether the client view must be loaded automatically when its data is accessed. (Inherited from C1.Data.ClientView<T>)
	Count	Gets the total number of elements in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	CurrentItem	Gets the current item in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	DeferredMaintenance	Gets the effective value of MaintenanceMode. (Inherited from C1.LiveLinq.LiveViews.View)
	FilterKey	Gets or sets a value that is used to filter items by comparing this value to the result of the filter key function applied to an item.
	FilterKeyType	Gets the filter key type. It is determined by the filter key function.
	IsLoaded	Gets a value indicating whether the client view is loaded. (Inherited from C1.Data.ClientView<T>)
	IsLoading	Gets a value indicating whether the client view is being loaded. (Inherited from C1.Data.ClientView<T>)

	IsReadOnly	Gets a value indicating whether this view is read-only, not updatable. (Inherited from C1.LiveLinq.LiveViews.View)
	Item	Gets the view item (element) at the specified ordinal position. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	MaintenanceMode	Gets or sets a value controlling how the view is synchronized with changes in its base data. (Inherited from C1.LiveLinq.LiveViews.View)
	MoveToFirstOnReset	Gets or sets a value indicating that the first item must be made current after initial loading or reset (on any C1.LiveLinq.SourceChangeType.Reset notification) if current item was not set by other means. The default is True. (Inherited from C1.LiveLinq.LiveViews.View)
	Operator	Gets a System.Windows.Controls.FilterOperator used to compare filter keys.
	Order	Gets a value indicating whether and how this view preserves item order if it exists in its base data source. (Inherited from C1.LiveLinq.LiveViews.View)
	Scope	Gets the client scope to which this client view belongs. (Inherited from C1.Data.ClientView<T>)
	Transaction	Gets an instance of C1.LiveLinq.ITransaction associated with the view. If a view has a transaction associated with it, that transaction's scope is opened automatically every time the view is updated, so the programmer does not need to do it manually in code. (Inherited from C1.LiveLinq.LiveViews.View)

Top

See Also

Reference

FilteredView<T> Class
C1.Data Namespace

FilterKey Property
Gets or sets a value that is used to filter items by comparing this value to the result of the filter key function applied to an item.

Syntax

Visual Basic (Declaration)
`Public Property FilterKey As Object`
C#
`public object FilterKey {get; set;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

FilteredView<T> Class
FilteredView<T> Members
Operator Property

FilterKeyType Property
Gets the filter key type. It is determined by the filter key function.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property FilterKeyType As Type`
C#
`public Type FilterKeyType {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

FilteredView<T> Class
FilteredView<T> Members

Operator Property
Gets a System.Windows.Controls.FilterOperator used to compare filter keys.

Syntax

Visual Basic (Declaration)
`Public Property Operator As FilterOperator`
C#
`public FilterOperator Operator {get; set;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

FilteredView<T> Class
FilteredView<T> Members
FilterOperator Enumeration

Fields
>

Name	Description
S Unfiltered	A special value indicating that filtering must not be performed. To disable

filtering, assign the value of this field to the FilterKey property.

Top

See Also

Reference

FilteredView<T> Class
C1.Data Namespace

Unfiltered Field
A special value indicating that filtering must not be performed. To disable filtering, assign the value of this field to the FilterKey property.

Syntax

Visual Basic (Declaration)
`Public Shared ReadOnly Unfiltered As Object`
C#
`public static readonly object Unfiltered`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

FilteredView<T> Class
FilteredView<T> Members

PagingView<T>
The type of the entities in the client view.

Represents a paged client view.

Object Model

PagingView<T>

Syntax

Visual Basic (Declaration)

`Public Class PagingView(Of T)`
`  Inherits ClientView(Of T)`
`  Implements C1.LiveLinq.IObservableSource(Of T)`

C#
`public class PagingView<T> : ClientView<T>, C1.LiveLinq.IObservableSource<T>`

Type Parameters

T

The type of the entities in the client view.

Remarks

Paging is performed on the server. It is controled by the PageSize and PageIndex properties.

Sorting is required, paging entities is impossible without sort.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.View
    C1.LiveLinq.LiveViews.View<T>
      C1.Data.ClientView<T>
        C1.Data.PagingView<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

PagingView<T> Members
C1.Data Namespace

Overview
The type of the entities in the client view.

Represents a paged client view.

Object Model

Syntax

Visual Basic (Declaration)

`Public Class PagingView(Of T)`
`  Inherits ClientView(Of T)`
`  Implements C1.LiveLinq.IObservableSource(Of T)`

C#
`public class PagingView<T> : ClientView<T>, C1.LiveLinq.IObservableSource<T>`

Type Parameters

T

The type of the entities in the client view.

Remarks

Paging is performed on the server. It is controled by the PageSize and PageIndex properties.

Sorting is required, paging entities is impossible without sort.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.View
    C1.LiveLinq.LiveViews.View<T>
      C1.Data.ClientView<T>
        C1.Data.PagingView<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

PagingView<T> Members
C1.Data Namespace

Members
Properties Methods Events

The following tables list the members exposed by PagingView<T>.

Public Properties

	Name	Description
	AutoLoad	Gets or sets a boolean value indicating whether the client view must be loaded automatically when its data is accessed. (Inherited from C1.Data.ClientView<T>)
	Count	Gets the total number of elements in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	CurrentItem	Gets the current item in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	DeferredMaintenance	Gets the effective value of MaintenanceMode. (Inherited from C1.LiveLinq.LiveViews.View)
	IsLoaded	Gets a value indicating whether the client view is loaded. (Inherited from C1.Data.ClientView<T>)
	IsLoading	Gets a value indicating whether the client view is being loaded.

		(Inherited from C1.Data.ClientView<T>)
	IsReadOnly	Gets a value indicating whether this view is read-only, not updatable. (Inherited from C1.LiveLinq.LiveViews.View)
	Item	Gets the view item (element) at the specified ordinal position. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	LoadSize	Gets or sets a value controling the number of entities to load in one batch.
	MaintenanceMode	Gets or sets a value controlling how the view is synchronized with changes in its base data. (Inherited from C1.LiveLinq.LiveViews.View)
	MoveToFirstOnReset	Gets or sets a value indicating that the first item must be made current after initial loading or reset (on any C1.LiveLinq.SourceChangeType.Reset notification) if current item was not set by other means. The default is True. (Inherited from C1.LiveLinq.LiveViews.View)
	Order	Gets a value indicating whether and how this view preserves item order if it exists in its base data source. (Inherited from C1.LiveLinq.LiveViews.View)
	PageCount	Gets the number of pages in this view.
	PageIndex	Gets or sets the index of the current page.
	PageSize	Gets or sets the number of items in a page.
	Scope	Gets the client scope to which this client view belongs. (Inherited from C1.Data.ClientView<T>)
	TotalItemCount	Gets the total number of entities in the view before paging is applied.

	Transaction	Gets an instance of C1.LiveLinq.ITransaction associated with the view. If a view has a transaction associated with it, that transaction's scope is opened automatically every time the view is updated, so the programmer does not need to do it manually in code. (Inherited from C1.LiveLinq.LiveViews.View)

Top

Public Methods

	Name	Description
	AsCollectionViewFactory	Returns an instance of System.ComponentModel.ICollectionViewFactory that can be used as a source of a System.Windows.Data.CollectionViewSource. (Inherited from C1.LiveLinq.LiveViews.View)
	AsDynamic	Used for views with anonymous type constructor as the result selector, converts the C1.LiveLinq.LiveViews.View to a View<dynamic> so it can be used for data binding and programmatic access. (Inherited from C1.LiveLinq.LiveViews.View)
	AsFiltered	Filters the view on the server side using a predicate. (Inherited from C1.Data.ClientView<T>)
	AsFilteredBound<TKey>	Filters the view on the server using a key selector function and configurable value and operator. (Inherited from C1.Data.ClientView<T>)
	AttachAggregationView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	AttachView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	CancelLoad	Cancels the current loading operation. (Inherited from C1.Data.ClientView<T>)

	Concat	Overloaded. Concatenation of two views. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Contains	Determines whether the view contains a specified item. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	DeferMaintenance	Enters a defer cycle that you can use to make bulk changes to the view sources and delay automatic view maintenance. (Inherited from C1.LiveLinq.LiveViews.View)
	GetEnumerator	Returns an enumerator that iterates through the view items. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	GroupBy	Overloaded. Groups the elements of a view according to a specified key selector function. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	GroupJoin	Overloaded. Correlates the elements of two views based on equality of keys and groups the results. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	IndexOf	Searches for the specified object among the elements of the view and returns the zero-based ordinal position of its first occurrence. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Join	Overloaded. Correlates the elements of two views based on matching keys. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Load	Loads the entities of the client view. (Inherited from C1.Data.ClientView<T>)
	Maintain	Brings the view up to date with its source data. (Inherited from C1.LiveLinq.LiveViews.View)

	OrderBy<TKey>	Sorts the elements of a view in ascending order. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	OrderByDescending<TKey>	Sorts the elements of a view in descending order. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Paging	Overloaded. Applies paging to this client view. (Inherited from C1.Data.ClientView<T>)
	ProgressiveLoading	Overloaded. Specifies that the client view loading is performed not in a single trip to the server but in multiple batches, each loading a page of a limited size so the user sees the result and can interact with it before all pages are loaded. (Inherited from C1.Data.ClientView<T>)
	PurgeEmptyGroups	Remove empty groups from a grouping view. (Inherited from C1.LiveLinq.LiveViews.View)
	Rebuild	Re-populates the view by re-executing the view's query. (Inherited from C1.LiveLinq.LiveViews.View)
	Refresh	Loads the entities of the client view ignoring the client-side cache. (Inherited from C1.Data.ClientView<T>)
	Select<TResult>	Projects each element of a view into a new form. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	SelectMany	Overloaded. Projects each element of this view to a collection of collections, flattens the resulting collections into one collection, and invokes a result selector function on each element therein. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	SetTransaction	Sets the value of the Transaction property. (Inherited from C1.LiveLinq.LiveViews.View)

	ToString	Returns a string representing this view. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Union	Overloaded. Set union of two views. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Where	Filters the source view based on a predicate. (Inherited from C1.LiveLinq.LiveViews.View<T>)

Top

Public Events

	Name	Description
	Changed	Occurs after an item of the view or the entire view has changed. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Loaded	Occurs when the client view has finished loading succesfully, and also when an exception has been thrown during loading. (Inherited from C1.Data.ClientView<T>)

Top

See Also

Reference

PagingView<T> Class
C1.Data Namespace

Properties
For a list of all members of this type, see PagingView<T> members.

Public Properties

	Name	Description
	AutoLoad	Gets or sets a boolean value indicating whether the client view must be

		loaded automatically when its data is accessed. (Inherited from C1.Data.ClientView<T>)
	Count	Gets the total number of elements in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	CurrentItem	Gets the current item in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	DeferredMaintenance	Gets the effective value of MaintenanceMode. (Inherited from C1.LiveLinq.LiveViews.View)
	IsLoaded	Gets a value indicating whether the client view is loaded. (Inherited from C1.Data.ClientView<T>)
	IsLoading	Gets a value indicating whether the client view is being loaded. (Inherited from C1.Data.ClientView<T>)
	IsReadOnly	Gets a value indicating whether this view is read-only, not updatable. (Inherited from C1.LiveLinq.LiveViews.View)
	Item	Gets the view item (element) at the specified ordinal position. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	LoadSize	Gets or sets a value controling the number of entities to load in one batch.
	MaintenanceMode	Gets or sets a value controlling how the view is synchronized with changes in its base data. (Inherited from C1.LiveLinq.LiveViews.View)
	MoveToFirstOnReset	Gets or sets a value indicating that the first item must be made current after initial loading or reset (on any C1.LiveLinq.SourceChangeType.Reset notification) if current item was not set by other means. The default is True. (Inherited from

		C1.LiveLinq.LiveViews.View)
	Order	Gets a value indicating whether and how this view preserves item order if it exists in its base data source. (Inherited from C1.LiveLinq.LiveViews.View)
	PageCount	Gets the number of pages in this view.
	PageIndex	Gets or sets the index of the current page.
	PageSize	Gets or sets the number of items in a page.
	Scope	Gets the client scope to which this client view belongs. (Inherited from C1.Data.ClientView<T>)
	TotalItemCount	Gets the total number of entities in the view before paging is applied.
	Transaction	Gets an instance of C1.LiveLinq.ITransaction associated with the view. If a view has a transaction associated with it, that transaction's scope is opened automatically every time the view is updated, so the programmer does not need to do it manually in code. (Inherited from C1.LiveLinq.LiveViews.View)

Top

See Also

Reference

PagingView<T> Class
C1.Data Namespace

LoadSize Property
Gets or sets a value controling the number of entities to load in one batch.

Syntax

Visual Basic (Declaration)	

```
Public Property LoadSize As Integer
```

C#

```
public int LoadSize {get; set;}
```

Remarks

Entities will be loaded using the multiple of PageSize nearest LoadSize. This allows multiple pages to be loaded at once without loading partial pages.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

PagingView<T> Class
PagingView<T> Members

PageCount Property
Gets the number of pages in this view.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Property PageCount As Integer
```

C#

```
public int PageCount {get;}
```

Remarks

If PageSize is 0, PageCount is also 0.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

PagingView<T> Class
PagingView<T> Members

PageIndex Property
Gets or sets the index of the current page.

Syntax

Visual Basic (Declaration)
`Public Property PageIndex As Integer`
C#
`public int PageIndex {get; set;}`

Remarks

Setting this property value to an invalid value is ignored. A value is invalid if it is less than 0 or greater or equal to PageCount. If there are no items in this view, the only valid value for this property is 0.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

PagingView<T> Class
PagingView<T> Members

PageSize Property
Gets or sets the number of items in a page.

Syntax

Visual Basic (Declaration)
`Public Property PageSize As Integer`
C#
`public int PageSize {get; set;}`

Remarks

To disable paging, set this property to 0.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

PagingView<T> Class
PagingView<T> Members

TotalItemCount Property
Gets the total number of entities in the view before paging is applied.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property TotalItemCount As Integer`
C#
`public int TotalItemCount {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

PagingView<T> Class
PagingView<T> Members

ProgressiveView<T>
The type of the entities in this view.

Represents a client view that loads entities sequentially (progressively) page by page. The user sees the result and can interact with it before all pages are loaded.

Object Model

Syntax

Visual Basic (Declaration)

`Public Class ProgressiveView(Of T)`
`  Inherits ClientView(Of T)`
`  Implements C1.LiveLinq.IObservableSource(Of T)`

C#
`public class ProgressiveView<T> : ClientView<T>,` `C1.LiveLinq.IObservableSource<T>`

Type Parameters

T

The type of the entities in this view.

Remarks

Sorting is required, loading entities progressively is impossible without sort.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.View
    C1.LiveLinq.LiveViews.View<T>
      C1.Data.ClientView<T>
        C1.Data.ProgressiveView<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ProgressiveView<T> Members
C1.Data Namespace

Overview
The type of the entities in this view.

Represents a client view that loads entities sequentially (progressively) page by page. The user sees the result and can interact with it before all pages are loaded.

Object Model

Syntax

Visual Basic (Declaration)	

`Public Class ProgressiveView(Of T)`
`  Inherits ClientView(Of T)`
`  Implements C1.LiveLinq.IObservableSource(Of T)`	

C#	
`public class ProgressiveView<T> : ClientView<T>,`	

```
C1.LiveLinq.IObservableSource<T>
```

Type Parameters

T

The type of the entities in this view.

Remarks

Sorting is required, loading entities progressively is impossible without sort.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.View
    C1.LiveLinq.LiveViews.View<T>
      C1.Data.ClientView<T>
        C1.Data.ProgressiveView<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ProgressiveView<T> Members
C1.Data Namespace

Members
Properties Methods Events

The following tables list the members exposed by ProgressiveView<T>.

Public Properties

	Name	Description
	AutoLoad	Gets or sets a boolean value indicating whether the client view must be

		loaded automatically when its data is accessed. (Inherited from C1.Data.ClientView<T>)
	Count	Gets the total number of elements in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	CurrentItem	Gets the current item in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	DeferredMaintenance	Gets the effective value of MaintenanceMode. (Inherited from C1.LiveLinq.LiveViews.View)
	IsLoaded	Gets a value indicating whether the client view is loaded. (Inherited from C1.Data.ClientView<T>)
	IsLoading	Gets a value indicating whether the client view is being loaded. (Inherited from C1.Data.ClientView<T>)
	IsReadOnly	Gets a value indicating whether this view is read-only, not updatable. (Inherited from C1.LiveLinq.LiveViews.View)
	Item	Gets the view item (element) at the specified ordinal position. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	LoadSize	Gets or sets the size of a page. To disable progressive loading, set this property to 0.
	MaintenanceMode	Gets or sets a value controlling how the view is synchronized with changes in its base data. (Inherited from C1.LiveLinq.LiveViews.View)
	MoveToFirstOnReset	Gets or sets a value indicating that the first item must be made current after initial loading or reset (on any C1.LiveLinq.SourceChangeType.Reset notification) if current item was not set by other means. The default is True. (Inherited from

		C1.LiveLinq.LiveViews.View)
	Order	Gets a value indicating whether and how this view preserves item order if it exists in its base data source. (Inherited from C1.LiveLinq.LiveViews.View)
	Scope	Gets the client scope to which this client view belongs. (Inherited from C1.Data.ClientView<T>)
	Transaction	Gets an instance of C1.LiveLinq.ITransaction associated with the view. If a view has a transaction associated with it, that transaction's scope is opened automatically every time the view is updated, so the programmer does not need to do it manually in code. (Inherited from C1.LiveLinq.LiveViews.View)

Top

Public Methods

	Name	Description
	AsCollectionViewFactory	Returns an instance of System.ComponentModel.ICollectionViewFactory that can be used as a source of a System.Windows.Data.CollectionViewSource. (Inherited from C1.LiveLinq.LiveViews.View)
	AsDynamic	Used for views with anonymous type constructor as the result selector, converts the C1.LiveLinq.LiveViews.View to a View<dynamic> so it can be used for data binding and programmatic access. (Inherited from C1.LiveLinq.LiveViews.View)
	AsFiltered	Filters the view on the server side using a predicate. (Inherited from C1.Data.ClientView<T>)
	AsFilteredBound<TKey>	Filters the view on the server using a key selector function and configurable value and operator. (Inherited from

		C1.Data.ClientView<T>)
	AttachAggregationView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	AttachView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	CancelLoad	Cancels the current loading operation. (Inherited from C1.Data.ClientView<T>)
	Concat	Overloaded. Concatenation of two views. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Contains	Determines whether the view contains a specified item. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	DeferMaintenance	Enters a defer cycle that you can use to make bulk changes to the view sources and delay automatic view maintenance. (Inherited from C1.LiveLinq.LiveViews.View)
	GetEnumerator	Returns an enumerator that iterates through the view items. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	GroupBy	Overloaded. Groups the elements of a view according to a specified key selector function. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	GroupJoin	Overloaded. Correlates the elements of two views based on equality of keys and groups the results. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	IndexOf	Searches for the specified object among the elements of the view and returns the zero-based ordinal position of its first occurrence. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Join	Overloaded. Correlates the elements of two views based on

		matching keys. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Load	Loads the entities of the client view. (Inherited from C1.Data.ClientView<T>)
	Maintain	Brings the view up to date with its source data. (Inherited from C1.LiveLinq.LiveViews.View)
	OrderBy<TKey>	Sorts the elements of a view in ascending order. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	OrderByDescending<TKey>	Sorts the elements of a view in descending order. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Paging	Overloaded. Applies paging to this client view. (Inherited from C1.Data.ClientView<T>)
	ProgressiveLoading	Overloaded. Specifies that the client view loading is performed not in a single trip to the server but in multiple batches, each loading a page of a limited size so the user sees the result and can interact with it before all pages are loaded. (Inherited from C1.Data.ClientView<T>)
	PurgeEmptyGroups	Remove empty groups from a grouping view. (Inherited from C1.LiveLinq.LiveViews.View)
	Rebuild	Re-populates the view by re-executing the view's query. (Inherited from C1.LiveLinq.LiveViews.View)
	Refresh	Loads the entities of the client view ignoring the client-side cache. (Inherited from C1.Data.ClientView<T>)
	Select<TResult>	Projects each element of a view into a new form. (Inherited from C1.LiveLinq.LiveViews.View<T>)

	SelectMany	Overloaded. Projects each element of this view to a collection of collections, flattens the resulting collections into one collection, and invokes a result selector function on each element therein. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	SetTransaction	Sets the value of the Transaction property. (Inherited from C1.LiveLinq.LiveViews.View)
	ToString	Returns a string representing this view. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Union	Overloaded. Set union of two views. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Where	Filters the source view based on a predicate. (Inherited from C1.LiveLinq.LiveViews.View<T>)

Top

Public Events

	Name	Description
	Changed	Occurs after an item of the view or the entire view has changed. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Loaded	Occurs when the client view has finished loading succesfully, and also when an exception has been thrown during loading. (Inherited from C1.Data.ClientView<T>)

Top

See Also

Reference

ProgressiveView<T> Class
C1.Data Namespace

Properties

For a list of all members of this type, see ProgressiveView<T> members.

Public Properties

	Name	Description
	AutoLoad	Gets or sets a boolean value indicating whether the client view must be loaded automatically when its data is accessed. (Inherited from C1.Data.ClientView<T>)
	Count	Gets the total number of elements in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	CurrentItem	Gets the current item in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	DeferredMaintenance	Gets the effective value of MaintenanceMode. (Inherited from C1.LiveLinq.LiveViews.View)
	IsLoaded	Gets a value indicating whether the client view is loaded. (Inherited from C1.Data.ClientView<T>)
	IsLoading	Gets a value indicating whether the client view is being loaded. (Inherited from C1.Data.ClientView<T>)
	IsReadOnly	Gets a value indicating whether this view is read-only, not updatable. (Inherited from C1.LiveLinq.LiveViews.View)
	Item	Gets the view item (element) at the specified ordinal position. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	LoadSize	Gets or sets the size of a page. To disable progressive loading, set this property to 0.
	MaintenanceMode	Gets or sets a value controlling how the view is synchronized with

		changes in its base data. (Inherited from C1.LiveLinq.LiveViews.View)
	MoveToFirstOnReset	Gets or sets a value indicating that the first item must be made current after initial loading or reset (on any C1.LiveLinq.SourceChangeType.Reset notification) if current item was not set by other means. The default is True. (Inherited from C1.LiveLinq.LiveViews.View)
	Order	Gets a value indicating whether and how this view preserves item order if it exists in its base data source. (Inherited from C1.LiveLinq.LiveViews.View)
	Scope	Gets the client scope to which this client view belongs. (Inherited from C1.Data.ClientView<T>)
	Transaction	Gets an instance of C1.LiveLinq.ITransaction associated with the view. If a view has a transaction associated with it, that transaction's scope is opened automatically every time the view is updated, so the programmer does not need to do it manually in code. (Inherited from C1.LiveLinq.LiveViews.View)

Top

See Also

Reference

ProgressiveView<T> Class
C1.Data Namespace

LoadSize Property
Gets or sets the size of a page. To disable progressive loading, set this property to 0.

Syntax

Visual Basic (Declaration)
`Public Property LoadSize As Integer`

C#

```
public int LoadSize {get; set;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

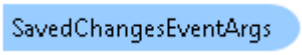
See Also

Reference

ProgressiveView<T> Class
ProgressiveView<T> Members

SavedChangesEventArgs
Provides data for the C1DataSource.SavedChanges event.

Object Model



Syntax

Visual Basic (Declaration)

```
Public Class SavedChangesEventArgs
   Inherits System.EventArgs
```

C#

```
public class SavedChangesEventArgs : System.EventArgs
```

Inheritance Hierarchy

System.Object
  System.EventArgs
    C1.Data.SavedChangesEventArgs

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

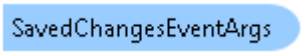
See Also

Reference

SavedChangesEventArgs Members
C1.Data Namespace

Overview
Provides data for the C1DataSource.SavedChanges event.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class SavedChangesEventArgs` `  Inherits System.EventArgs`
C#
`public class SavedChangesEventArgs : System.EventArgs`

Inheritance Hierarchy

System.Object
  System.EventArgs
    C1.Data.SavedChangesEventArgs

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SavedChangesEventArgs Members
C1.Data Namespace

Members
Properties Methods

The following tables list the members exposed by SavedChangesEventArgs.

Public Properties

	Name	Description
	Error	Gets a value showing the error that occurred during a save operation.
	HasError	Gets a value indicating whether the save operation has failed. If true, inspect the Error property for details.
	IsErrorHandled	Gets a value indicating whether the error has been marked as handled by calling MarkErrorAsHandled.

Top

Public Methods

	Name	Description
	MarkErrorAsHandled	If this method is called for a failed operation (if HasError is true), it marks the error as handled. Otherwise, an exception is thrown.

Top

See Also

Reference

SavedChangesEventArgs Class
C1.Data Namespace

Methods
For a list of all members of this type, see SavedChangesEventArgs members.

Public Methods

	Name	Description
	MarkErrorAsHandled	If this method is called for a failed operation (if HasError is true), it marks the error as handled. Otherwise, an exception is thrown.

Top

See Also

Reference

SavedChangesEventArgs Class
C1.Data Namespace

MarkErrorAsHandled Method
If this method is called for a failed operation (if HasError is true), it marks the error as handled. Otherwise, an exception is thrown.

Syntax

Visual Basic (Declaration)

```
Public Sub MarkErrorAsHandled()
```

C#

```
public void MarkErrorAsHandled()
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SavedChangesEventArgs Class
SavedChangesEventArgs Members

Properties
For a list of all members of this type, see SavedChangesEventArgs members.

Public Properties

	Name	Description
	Error	Gets a value showing the error that occurred during a save operation.
	HasError	Gets a value indicating whether the save operation has failed. If true, inspect the Error property for details.
	IsErrorHandled	Gets a value indicating whether the error has been marked as handled by calling MarkErrorAsHandled.

Top

See Also

Reference

SavedChangesEventArgs Class
C1.Data Namespace

Error Property
Gets a value showing the error that occurred during a save operation.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Property Error As Exception
```

C#

```
public Exception Error {get;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SavedChangesEventArgs Class
SavedChangesEventArgs Members

HasError Property
Gets a value indicating whether the save operation has failed. If true, inspect the Error property for details.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property HasError As Boolean`
C#
`public bool HasError {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SavedChangesEventArgs Class
SavedChangesEventArgs Members

IsErrorHandled Property
Gets a value indicating whether the error has been marked as handled by calling MarkErrorAsHandled.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property IsErrorHandled As Boolean`

C#

```
public bool IsErrorHandled {get;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SavedChangesEventArgs Class
SavedChangesEventArgs Members

C1.Data.DataSource Namespace

Overview

Classes

	Class	Description
	BaseControlHandler	A base class for control handlers that connect GUI controls of supported types to a C1DataSource so that those controls can be given additional functionality such as auto-lookup columns and virtual mode.
	ClientCollectionView	The collection view implementation used by a ClientViewSource and other Studio for Entity Framework data sources.
	ClientViewSource	Data source object exposing data from C1.Data.ClientCacheBase to which GUI controls can bind. Using a ClientViewSource, you can load, filter, group, and sort data easily.
	ClientViewSourceException	This exception indicates that a ClientViewSource is miconfigured or an error has occurred during the Load operation.

Enumerations

	Enumeration	Description
	VirtualModeKind	Enumeration of possible virtual modes a ClientViewSource can be in. Used in the VirtualMode property.

See Also

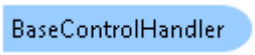
Reference

C1.Silverlight.Data.Entity Assembly

Classes

BaseControlHandler
A base class for control handlers that connect GUI controls of supported types to a C1DataSource so that those controls can be given additional functionality such as auto-lookup columns and virtual mode.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit Class BaseControlHandler` `  Inherits System.Windows.DependencyObject`
C#
`public abstract class BaseControlHandler : System.Windows.DependencyObject`

Remarks

Use platform-specific control handlers for your controls: C1.Win.Data.ControlHandler, C1.WPF.Data.ControlHandler, and C1.Silverlight.Data.ControlHandler.

The list of supported GUI controls for each platform can be found in the reference of that platform's ControlHandler.

Inheritance Hierarchy

System.Object
  System.Windows.DependencyObject
    C1.Data.DataSource.BaseControlHandler
      C1.Silverlight.Data.ControlHandler
      C1.Win.Data.ControlHandler
      C1.WPF.Data.ControlHandler

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

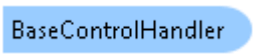
See Also

Reference

BaseControlHandler Members
C1.Data.DataSource Namespace

Overview
A base class for control handlers that connect GUI controls of supported types to a C1DataSource so that those controls can be given additional functionality such as auto-lookup columns and virtual mode.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit Class BaseControlHandler` `  Inherits System.Windows.DependencyObject`
C#
`public abstract class BaseControlHandler : System.Windows.DependencyObject`

Remarks

Use platform-specific control handlers for your controls: C1.Win.Data.ControlHandler, C1.WPF.Data.ControlHandler, and C1.Silverlight.Data.ControlHandler.

The list of supported GUI controls for each platform can be found in the reference of that platform's ControlHandler.

Inheritance Hierarchy

System.Object
  System.Windows.DependencyObject
    C1.Data.DataSource.BaseControlHandler
      C1.Silverlight.Data.ControlHandler
      C1.Win.Data.ControlHandler
      C1.WPF.Data.ControlHandler

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

BaseControlHandler Members
C1.Data.DataSource Namespace

Members
Fields Properties Methods

The following tables list the members exposed by BaseControlHandler.

Public Fields

	Name	Description
S	AutoLookupProperty	The DependencyProperty for the AutoLookup property.
S	VirtualModeProperty	The DependencyProperty for the VirtualMode property.

Top

Public Properties

	Name	Description
	AutoLookup	Gets or sets a value indicating whether data grid columns bound to navigation (foreign key, lookup) properties must be converted to combo box columns, so the user can see the right value and edit it by choosing a value from a drop-down list. The default value is False.
	Dispatcher	(Inherited from System.Windows.DependencyObject)
	SupportsVirtualMode	Gets a value indicating whether this control handler supports Virtual Mode.
	VirtualMode	Gets or sets a value indicating whether virtual mode specified in a ClientViewSource is managed by this control handler.

Top

Public Methods

	Name	Description
	Apply	Forces this control handler to apply its settings to the current control.
	ClearValue	(Inherited from System.Windows.DependencyObject)
	GetAnimationBaseValue	(Inherited from System.Windows.DependencyObject)
	GetValue	(Inherited from System.Windows.DependencyObject)
	ReadLocalValue	(Inherited from System.Windows.DependencyObject)
	SetValue	(Inherited from System.Windows.DependencyObject)

Top

See Also

Reference

BaseControlHandler Class
C1.Data.DataSource Namespace

Methods
For a list of all members of this type, see BaseControlHandler members.

Public Methods

	Name	Description
	Apply	Forces this control handler to apply its settings to the current control.
	ClearValue	(Inherited from System.Windows.DependencyObject)
	GetAnimationBaseValue	(Inherited from System.Windows.DependencyObject)
	GetValue	(Inherited from System.Windows.DependencyObject)
	ReadLocalValue	(Inherited from System.Windows.DependencyObject)
	SetValue	(Inherited from System.Windows.DependencyObject)

Top

See Also

Reference

BaseControlHandler Class
C1.Data.DataSource Namespace

Apply Method
Forces this control handler to apply its settings to the current control.

Syntax

Visual Basic (Declaration)
`Public Overridable Sub Apply()`
C#
`public virtual void Apply()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

BaseControlHandler Class
BaseControlHandler Members

Properties
For a list of all members of this type, see BaseControlHandler members.

Public Properties

	Name	Description
	AutoLookup	Gets or sets a value indicating whether data grid columns bound to navigation (foreign key, lookup) properties must be converted to combo box columns, so the user can see the right value and edit it by choosing a value from a drop-down list. The default value is False.
	Dispatcher	(Inherited from System.Windows.DependencyObject)
	SupportsVirtualMode	Gets a value indicating whether this control handler supports Virtual Mode.
	VirtualMode	Gets or sets a value indicating whether virtual mode specified in a ClientViewSource is managed by this control handler.

Top

See Also

Reference

BaseControlHandler Class
C1.Data.DataSource Namespace

AutoLookup Property
Gets or sets a value indicating whether data grid columns bound to navigation (foreign key, lookup) properties must be converted to combo box columns, so the user can see the right value and edit it by choosing a value from a drop-down list. The default value is False.

Syntax

Visual Basic (Declaration)
`Public Property AutoLookup As Boolean`
C#
`public bool AutoLookup {get; set;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

BaseControlHandler Class
BaseControlHandler Members

SupportsVirtualMode Property
Gets a value indicating whether this control handler supports Virtual Mode.

Syntax

Visual Basic (Declaration)

```
Public Overridable ReadOnly Property SupportsVirtualMode As Boolean
```

C#

```
public virtual bool SupportsVirtualMode {get;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

BaseControlHandler Class
BaseControlHandler Members
VirtualMode Property

VirtualMode Property
Gets or sets a value indicating whether virtual mode specified in a ClientViewSource is managed by this control handler.

Syntax

Visual Basic (Declaration)

```
Public Property VirtualMode As Boolean
```

C#

```
public bool VirtualMode {get; set;}
```

Remarks

Setting this property to True has effect only if VirtualMode of the associated ClientViewSource is set to Managed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

BaseControlHandler Class
BaseControlHandler Members

Fields
For a list of all members of this type, see BaseControlHandler members.

Public Fields

	Name	Description
S	AutoLookupProperty	The DependencyProperty for the AutoLookup property.
S	VirtualModeProperty	The DependencyProperty for the VirtualMode property.

Top

See Also

Reference

BaseControlHandler Class
C1.Data.DataSource Namespace

AutoLookupProperty Field
The DependencyProperty for the AutoLookup property.

Syntax

Visual Basic (Declaration)

```
Public Shared ReadOnly AutoLookupProperty As DependencyProperty
```

C#

```
public static readonly DependencyProperty AutoLookupProperty
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

BaseControlHandler Class
BaseControlHandler Members

VirtualModeProperty Field
The DependencyProperty for the VirtualMode property.

Syntax

Visual Basic (Declaration)
`Public Shared ReadOnly VirtualModeProperty As DependencyProperty`
C#
`public static readonly DependencyProperty VirtualModeProperty`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

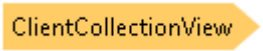
See Also

Reference

BaseControlHandler Class
BaseControlHandler Members

ClientCollectionView
The collection view implementation used by a ClientViewSource and other Studio for Entity Framework data sources.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class ClientCollectionView`
C#
`public class ClientCollectionView`

Inheritance Hierarchy

System.Object
  C1.Data.DataSource.ClientCollectionView

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Members
C1.Data.DataSource Namespace

Overview
The collection view implementation used by a ClientViewSource and other Studio for Entity Framework data sources.

Object Model

Syntax

Visual Basic (Declaration)
`Public Class ClientCollectionView`

C#
`public class ClientCollectionView`

Inheritance Hierarchy

System.Object
  C1.Data.DataSource.ClientCollectionView

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Members
C1.Data.DataSource Namespace

Members
Properties Methods Events

The following tables list the members exposed by ClientCollectionView.

Public Properties

	Name	Description
	CanAdd	Gets a value indicating whether the Add method is supported.
	CanChangePage	Gets a value that indicates whether the PageIndex value can change.
	CanRemove	Gets a value that indicates whether an item can be removed from the collection.
	CollectionViewFactory	Gets an instance of System.ComponentModel.ICollectionViewFactory that can be used as a source of a System.Windows.Data.CollectionViewSource.

	Count	Gets the number of elements contained in the ClientCollectionView.
	CurrentItem	Gets the current item in the view.
	CurrentPosition	Gets the ordinal position of the CurrentItem within the collection view.
	IsEmpty	Returns a value that indicates whether the resulting view is empty.
	IsPageChanging	Gets a value that indicates whether the page index is changing.
	Item	Gets the element at the specified index.
	PageCount	Gets the count of the pages in this view.
	PageIndex	Gets the zero-based index of the current page.
	PageSize	Gets or sets the number of items to display on a page.
	TotalItemCount	Gets the total number of items in the view before paging is applied.

Top

Public Methods

	Name	Description
	Add	Adds a new entity to the client-side cache and to the associated context. The entity will appear in this collection view if it matches the underlying query.
	AsLive<T>	Converts this ClientCollectionView to a live view.
	Contains	Returns a value that indicates whether a given item belongs to this collection view.
	IndexOf	Determines the index of a specific item in the ClientCollectionView.

	MoveCurrentTo	Sets the specified item to be the CurrentItem in the view.
	MoveCurrentToFirst	Sets the first item in the view as the CurrentItem.
	MoveCurrentToLast	Sets the last item in the view as the CurrentItem.
	MoveCurrentToNext	Sets the item after the CurrentItem in the view as the System.ComponentModel.ICollectionView.CurrentItem.
	MoveCurrentToPosition	Sets the item at the specified index to be the CurrentItem in the view.
	MoveCurrentToPrevious	Sets the item before the CurrentItem in the view as the CurrentItem.
	MoveToFirstPage	Sets the first page as the current page.
	MoveToLastPage	Sets the last page as the current page.
	MoveToNextPage	Moves to the page after the current page.
	MoveToPage	Sets the first page as the current page.
	MoveToPreviousPage	Moves to the page before the current page.
	Remove	Removes the specified item from the collection.
	RemoveAt	Removes the item at the specified position from the collection.

Top

Public Events

	Name	Description
	CurrentChanged	When implementing this interface, raise this event after the current item has been changed.

	CurrentChanging	When implementing this interface, raise this event before changing the current item. Event handler can cancel this event.
	PageChanged	Occurs after the PageIndex has changed.
	PageChanging	Occurs before changing the PageIndex.
	PropertyChanged	Occurs when a property value changes.

Top

See Also

Reference

ClientCollectionView Class
C1.Data.DataSource Namespace

Methods
For a list of all members of this type, see ClientCollectionView members.

Public Methods

	Name	Description
	Add	Adds a new entity to the client-side cache and to the associated context. The entity will appear in this collection view if it matches the underlying query.
	AsLive<T>	Converts this ClientCollectionView to a live view.
	Contains	Returns a value that indicates whether a given item belongs to this collection view.
	IndexOf	Determines the index of a specific item in the ClientCollectionView.
	MoveCurrentTo	Sets the specified item to be the CurrentItem in the view.

	MoveCurrentToFirst	Sets the first item in the view as the CurrentItem.
	MoveCurrentToLast	Sets the last item in the view as the CurrentItem.
	MoveCurrentToNext	Sets the item after the CurrentItem in the view as the System.ComponentModel.ICollectionView.CurrentItem.
	MoveCurrentToPosition	Sets the item at the specified index to be the CurrentItem in the view.
	MoveCurrentToPrevious	Sets the item before the CurrentItem in the view as the CurrentItem.
	MoveToFirstPage	Sets the first page as the current page.
	MoveToLastPage	Sets the last page as the current page.
	MoveToNextPage	Moves to the page after the current page.
	MoveToPage	Sets the first page as the current page.
	MoveToPreviousPage	Moves to the page before the current page.
	Remove	Removes the specified item from the collection.
	RemoveAt	Removes the item at the specified position from the collection.

Top

See Also

Reference

ClientCollectionView Class
C1.Data.DataSource Namespace

Add Method
The new entity to add.

Adds a new entity to the client-side cache and to the associated context. The entity will appear in this collection view if it matches the underlying query.

Syntax

Visual Basic (Declaration)

```
Public Sub Add( _
   ByVal entity As Object _
)
```

C#

```
public void Add(
   object entity
)
```

Parameters

entity

The new entity to add.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

AsLive<T> Method
The type of the elements in this collection view.

Converts this ClientCollectionView to a live view.

Syntax

Visual Basic (Declaration)

```
Public Function AsLive(Of T)() As View(Of T)
```

C#

```
public View<T> AsLive<T>()
```

Type Parameters

T

The type of the elements in this collection view.

Return Value

The resulting live view.

Exceptions

Exception	Description
System.NotSupportedException	The ClientViewSource is in virtual mode.

Remarks

This method does not change the ClientCollectionView in any way, it just exposes its live view functionality.

This method is not supported for a ClientViewSource in virtual mode.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

Contains Method
The object to check.

Returns a value that indicates whether a given item belongs to this collection view.

Syntax

Visual Basic (Declaration)

`Public Function Contains( _`
`  ByVal item As Object _`
`) As Boolean`

C#

`public bool Contains(`
`  object item`
`)`

Parameters

item

The object to check.

Return Value

true if the item belongs to this collection view; otherwise, false.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

IndexOf Method
The item to locate in the ClientCollectionView.

Determines the index of a specific item in the ClientCollectionView.

Syntax

Visual Basic (Declaration)

```
Public Function IndexOf( _
   ByVal item As Object _
) As Integer
```

C#

```
public int IndexOf(
   object item
)
```

Parameters

item

The item to locate in the ClientCollectionView.

Return Value

The index of the item if found in the list; otherwise, -1.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

MoveCurrentTo Method
The item to set as the CurrentItem.

Sets the specified item to be the CurrentItem in the view.

Syntax

Visual Basic (Declaration)

```
Public Function MoveCurrentTo( _
```

```
   ByVal item As Object _
) As Boolean
```

C#

```
public bool MoveCurrentTo(
   object item
)
```

Parameters

item

The item to set as the CurrentItem.

Return Value

true if the resulting CurrentItem is within the view; otherwise, false.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

MoveCurrentToFirst Method

Sets the first item in the view as the CurrentItem.

Syntax

Visual Basic (Declaration)

```
Public Function MoveCurrentToFirst() As Boolean
```

C#

```
public bool MoveCurrentToFirst()
```

Return Value

true if the resulting CurrentItem is an item within the view; otherwise, false.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

MoveCurrentToLast Method
Sets the last item in the view as the CurrentItem.

Syntax

Visual Basic (Declaration)
`Public Function MoveCurrentToLast() As Boolean`
C#
`public bool MoveCurrentToLast()`

Return Value

true if the resulting CurrentItem is an item within the view; otherwise, false.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

MoveCurrentToNext Method

Sets the item after the CurrentItem in the view as the System.ComponentModel.ICollectionView.CurrentItem.

Syntax

Visual Basic (Declaration)
`Public Function MoveCurrentToNext() As Boolean`
C#
`public bool MoveCurrentToNext()`

Return Value

true if the resulting CurrentItem is an item within the view; otherwise, false.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

MoveCurrentToPosition Method

The index to set the System.ComponentModel.ICollectionView.CurrentItem to.

Sets the item at the specified index to be the CurrentItem in the view.

Syntax

Visual Basic (Declaration)

`Public Function MoveCurrentToPosition( _`
`  ByVal position As Integer _`
`) As Boolean`

C#

```
public bool MoveCurrentToPosition(
   int position
)
```

Parameters

position

The index to set the System.ComponentModel.ICollectionView.CurrentItem to.

Return Value

true if the resulting CurrentItem is an item within the view; otherwise, false.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

MoveCurrentToPrevious Method

Sets the item before the CurrentItem in the view as the CurrentItem.

Syntax

Visual Basic (Declaration)

```
Public Function MoveCurrentToPrevious() As Boolean
```

C#

```
public bool MoveCurrentToPrevious()
```

Return Value

true if the resulting CurrentItem is an item within the view; otherwise, false.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

MoveToFirstPage Method
Sets the first page as the current page.

Syntax

Visual Basic (Declaration)
`Public Function MoveToFirstPage() As Boolean`
C#
`public bool MoveToFirstPage()`

Return Value

true if the operation was successful; otherwise, false.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

MoveToLastPage Method
Sets the last page as the current page.

Syntax

Visual Basic (Declaration)
`Public Function MoveToLastPage() As Boolean`
C#
`public bool MoveToLastPage()`

Return Value

true if the operation was successful; otherwise, false.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

MoveToNextPage Method
Moves to the page after the current page.

Syntax

Visual Basic (Declaration)
`Public Function MoveToNextPage() As Boolean`
C#
`public bool MoveToNextPage()`

Return Value

true if the operation was successful; otherwise, false.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

MoveToPage Method
The index of the page to move to.

Sets the first page as the current page.

Syntax

Visual Basic (Declaration)

`Public Function MoveToPage( _`
`  ByVal pageIndex As Integer _`
`) As Boolean`

C#

`public bool MoveToPage(`
`  int pageIndex`
`)`

Parameters

pageIndex

The index of the page to move to.

Return Value

True if the operation was successful; otherwise, False.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

MoveToPreviousPage Method
Moves to the page before the current page.

Syntax

Visual Basic (Declaration)
`Public Function MoveToPreviousPage() As Boolean`
C#
`public bool MoveToPreviousPage()`

Return Value

true if the operation was successful; otherwise, false.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

Remove Method
The item to remove.

Removes the specified item from the collection.

Syntax

Visual Basic (Declaration)

```
Public Sub Remove( _
   ByVal item As Object _
)
```

C#

```
public void Remove(
   object item
)
```

Parameters

item

The item to remove.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

RemoveAt Method
The position of the item to remove.

Removes the item at the specified position from the collection.

Syntax

Visual Basic (Declaration)

```
Public Sub RemoveAt( _
   ByVal index As Integer _
)
```

C#

```
public void RemoveAt(
   int index
)
```

Parameters

index

The position of the item to remove.

Exceptions

Exception	Description
System.ArgumentOutOfRangeException	index is less than 0 or greater than the number of items in the collection view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

Properties
For a list of all members of this type, see ClientCollectionView members.

Public Properties

	Name	Description
	CanAdd	Gets a value indicating whether the Add method is supported.

	CanChangePage	Gets a value that indicates whether the PageIndex value can change.
	CanRemove	Gets a value that indicates whether an item can be removed from the collection.
	CollectionViewFactory	Gets an instance of System.ComponentModel.ICollectionViewFactory that can be used as a source of a System.Windows.Data.CollectionViewSource.
	Count	Gets the number of elements contained in the ClientCollectionView.
	CurrentItem	Gets the current item in the view.
	CurrentPosition	Gets the ordinal position of the CurrentItem within the collection view.
	IsEmpty	Returns a value that indicates whether the resulting view is empty.
	IsPageChanging	Gets a value that indicates whether the page index is changing.
	Item	Gets the element at the specified index.
	PageCount	Gets the count of the pages in this view.
	PageIndex	Gets the zero-based index of the current page.
	PageSize	Gets or sets the number of items to display on a page.
	TotalItemCount	Gets the total number of items in the view before paging is applied.

Top

See Also

Reference

ClientCollectionView Class
C1.Data.DataSource Namespace

CanAdd Property

Gets a value indicating whether the Add method is supported.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property CanAdd As Boolean`
C#
`public bool CanAdd {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

CanChangePage Property

Gets a value that indicates whether the PageIndex value can change.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property CanChangePage As Boolean`
C#
`public bool CanChangePage {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

CanRemove Property
Gets a value that indicates whether an item can be removed from the collection.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property CanRemove As Boolean`
C#
`public bool CanRemove {get;}`

Property Value

true if an item can be removed from the collection; otherwise, false.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

CollectionViewFactory Property
Gets an instance of System.ComponentModel.ICollectionViewFactory that can be used as a source of a System.Windows.Data.CollectionViewSource.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Property CollectionViewFactory As ICollectionViewFactory
```

C#

```
public ICollectionViewFactory CollectionViewFactory {get;}
```

Property Value

A factory that returns the same ClientCollectionView as an System.ComponentModel.ICollectionView.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

Count Property
Gets the number of elements contained in the ClientCollectionView.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Property Count As Integer
```

C#

```
public int Count {get;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

CurrentItem Property
Gets the current item in the view.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property CurrentItem As Object`
C#
`public object CurrentItem {get;}`

Property Value

The current item of the view or null if there is no current item.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

CurrentPosition Property
Gets the ordinal position of the CurrentItem within the collection view.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property CurrentPosition As Integer`

C#

```
public int CurrentPosition {get;}
```

Property Value

The ordinal position of the CurrentItem within the collection view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

IsEmpty Property
Returns a value that indicates whether the resulting view is empty.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Property IsEmpty As Boolean
```

C#

```
public bool IsEmpty {get;}
```

Property Value

true if the resulting view is empty; otherwise, false.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

IsPageChanging Property
Gets a value that indicates whether the page index is changing.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Property IsPageChanging As Boolean
```

C#

```
public bool IsPageChanging {get;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

Item Property
The zero-based index of the element to get.

Gets the element at the specified index.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Default Property Item( _
   ByVal index As Integer _
) As Object
```

C#

```
public object this[
   int index
]; {get;}
```

Parameters

index

The zero-based index of the element to get.

Property Value

The element at the specified index.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

PageCount Property
Gets the count of the pages in this view.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Property PageCount As Integer
```

C#

```
public int PageCount {get;}
```

Remarks

When PageSize is 0, the PageIndex will also be 0.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

PageIndex Property
Gets the zero-based index of the current page.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property PageIndex As Integer`
C#
`public int PageIndex {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

PageSize Property
Gets or sets the number of items to display on a page.

Syntax

Visual Basic (Declaration)
`Public Property PageSize As Integer`
C#
`public int PageSize {get; set;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

TotalItemCount Property
Gets the total number of items in the view before paging is applied.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property TotalItemCount As Integer`
C#
`public int TotalItemCount {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

Events
>

Name	Description
CurrentChanged	When implementing this interface, raise this event after the current item has been changed.
CurrentChanging	When implementing this interface, raise this event before changing the current item. Event handler can cancel this event.
PageChanged	Occurs after the PageIndex has changed.
PageChanging	Occurs before changing the PageIndex.
PropertyChanged	Occurs when a property value changes.

Top

See Also

Reference

ClientCollectionView Class
C1.Data.DataSource Namespace

CurrentChanged Event
When implementing this interface, raise this event after the current item has been changed.

Syntax

Visual Basic (Declaration)

```
Public Event CurrentChanged As EventHandler
```

C#

```
public event EventHandler CurrentChanged
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

CurrentChanging Event
When implementing this interface, raise this event before changing the current item. Event handler can cancel this event.

Syntax

Visual Basic (Declaration)	
`Public Event CurrentChanging As CurrentChangingEventHandler`	
C#	
`public event CurrentChangingEventHandler CurrentChanging`	

Event Data

The event handler receives an argument of type CurrentChangingEventArgs containing data related to this event. The following **CurrentChangingEventArgs** properties provide information specific to this event.

Property	Description
Cancel	Gets or sets a value that indicates whether to cancel the event.
IsCancelable	Gets a value that indicates whether the event is cancelable.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

PageChanged Event
Occurs after the PageIndex has changed.

Syntax

Visual Basic (Declaration)
`Public Event PageChanged As EventHandler(Of EventArgs)`
C#
`public event EventHandler<EventArgs> PageChanged`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

PageChanging Event
Occurs before changing the PageIndex.

Syntax

Visual Basic (Declaration)
`Public Event PageChanging As EventHandler(Of PageChangingEventArgs)`
C#

```
public event EventHandler<PageChangingEventArgs> PageChanging
```

Event Data

The event handler receives an argument of type PageChangingEventArgs containing data related to this event. The following **PageChangingEventArgs** properties provide information specific to this event.

Property	Description
Cancel (Inherited from System.ComponentModel.CancelEventArgs)	
NewPageIndex	

Remarks

This event allows to cancel the ongoing PageIndex change by setting System.ComponentModel.CancelEventArgs.Cancel to true.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

PropertyChanged Event
Occurs when a property value changes.

Syntax

Visual Basic (Declaration)	
`Public Event PropertyChanged As PropertyChangedEventHandler`	
C#	

```
public event PropertyChangedEventHandler PropertyChanged
```

Event Data

The event handler receives an argument of type PropertyChangedEventArgs containing data related to this event. The following **PropertyChangedEventArgs** properties provide information specific to this event.

Property	Description
PropertyName	

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientCollectionView Class
ClientCollectionView Members

ClientViewSource
Data source object exposing data from C1.Data.ClientCacheBase to which GUI controls can bind. Using a ClientViewSource, you can load, filter, group, and sort data easily.

Object Model

Syntax

Visual Basic (Declaration)	
`Public Class ClientViewSource` `  Inherits System.Windows.DependencyObject`	
C#	

```
public class ClientViewSource : System.Windows.DependencyObject
```

Inheritance Hierarchy

System.Object
  System.Windows.DependencyObject
    C1.Data.DataSource.ClientViewSource
      C1.Data.Entities.EntityViewSource
      C1.Silverlight.Data.RiaServices.RiaViewSource

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

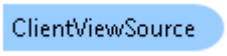
See Also

Reference

ClientViewSource Members
C1.Data.DataSource Namespace

Overview
Data source object exposing data from C1.Data.ClientCacheBase to which GUI controls can bind. Using a ClientViewSource, you can load, filter, group, and sort data easily.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class ClientViewSource` `  Inherits System.Windows.DependencyObject`
C#
`public class ClientViewSource : System.Windows.DependencyObject`

Inheritance Hierarchy

System.Object
  System.Windows.DependencyObject
    C1.Data.DataSource.ClientViewSource
      C1.Data.Entities.EntityViewSource
      C1.Silverlight.Data.RiaServices.RiaViewSource

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Members
C1.Data.DataSource Namespace

Members
Properties Methods Events

The following tables list the members exposed by ClientViewSource.

Public Constructors

	Name	Description
	ClientViewSource Constructor	Initializes a new instance of the ClientViewSource class.

Top

Public Properties

	Name	Description
	AutoLoad	Gets or sets a value indicating whether Load is automatically invoked on startup and when a change occurs that impacts the query composed by the ClientViewSource. The default is True.
	BaseView	Gets or sets an instance of C1.Data.ClientView<T> that the

		ClientViewSource uses as the base for composing queries.
	CacheTimeout	Gets or sets the period of time entities loaded in virtual mode are kept in the cache without checking whether they are needed or not. If an entity was neither used nor considered needed for a period of time longer than CacheTimeout, ClientViewSource may evict it from the cache.
	CurrentClientView	Gets the current client view used to load entities, or null in virtual mode.
	DataView	Gets the current view of entities resulting from the last load operation.
	Dispatcher	(Inherited from System.Windows.DependencyObject)
	FilterDescriptors	Gets the collection of System.Windows.Controls.FilterDescriptor objects used when performing loads.
	FilterOperator	Gets or sets the logical operator used for combinining FilterDescriptors in the filter collection. The default value is System.Windows.Controls.FilterDescriptorLogicalOperator.And.
	GroupDescriptors	Gets the collection of System.Windows.Controls.GroupDescriptor objects used to organize the loaded entities into groups.
	IsLoadingData	Gets a value indicating whether the ClientViewSource is currently loading data.
	LoadCommand	Gets an System.Windows.Input.ICommand that invokes Load on this ClientViewSource.
	LoadDelay	Gets or sets the delay before an automatic data loading operation is started. It is the delay from the time a change prompting automatic load occurs until the time the resulting Load is started. The default delay is 25 milliseconds.
	LoadSize	Gets or sets the maximum number of items to load each time a Load is

		executed. When equal to 0, all requested entities will be loaded. The default is 0.
	MoveToFirstOnLoad	Gets or sets a value indicating that the first item must be made current after Load operation is completed if current item was not set by other means.
	Name	Gets a name of this ClientViewSource to reference it in a C1DataSource.ViewSources collection. By default it is determined by the EntitySetName (for Entity Framework) or QueryName (for RIA Services), but it can be overrided by NameOverride.
	NameOverride	Gets or sets a value that overrides the value of the Name property.
	PageSize	Gets or sets the number of items displayed on each page of the DataView, or the number of items to fetch in each query in virtual mode, or 0 to disable paging.
	SortDescriptors	Gets the collection of System.Windows.Controls.SortDescriptor objects used to sort the data.
	VirtualMode	Gets or sets a value indicating whether the ClientViewSource is in virtual mode. Virtual mode is an innovative technology allowing to bind GUI controls directly to very large data sets without delays and performance degradation and without inconvenience of paging. By default, virtual mode is disabled (the default value is VirtualModeKind.None).

Top

Public Methods

	Name	Description
	ClearValue	(Inherited from System.Windows.DependencyObject)

	DeferLoad	Used to group changes to multiple load-affecting properties together, deferring the resulting load operations so a single load operation is performed in the end, that is, when the object returned from this method is disposed.
	GetAnimationBaseValue	(Inherited from System.Windows.DependencyObject)
	GetValue	(Inherited from System.Windows.DependencyObject)
	Load	Starts a load operation. Any pending load will be implicitly canceled.
	LoadRange	If in virtual mode, loads a specific range of entities.
	ReadLocalValue	(Inherited from System.Windows.DependencyObject)
	Refresh	Starts a load operation ignoring the client-side cache. Any pending load will be implicitly canceled.
	SetValue	(Inherited from System.Windows.DependencyObject)

Top

Public Events

	Name	Description
	LoadedData	Occurs when a load operation is completed, or when an exception was thrown during the load operation.
	PropertyChanged	Occurs when a property value changes.

Top

See Also

Reference

ClientViewSource Class
C1.Data.DataSource Namespace

ClientViewSource Constructor
Initializes a new instance of the ClientViewSource class.

Syntax

Visual Basic (Declaration)
`Public Function New()`
C#
`public ClientViewSource()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

Methods
For a list of all members of this type, see ClientViewSource members.

Public Methods

	Name	Description
	ClearValue	(Inherited from System.Windows.DependencyObject)
	DeferLoad	Used to group changes to multiple load-affecting properties together, deferring the resulting load operations so a single load operation is performed in the end, that is, when the object returned from this method is disposed.

	GetAnimationBaseValue	(Inherited from System.Windows.DependencyObject)
	GetValue	(Inherited from System.Windows.DependencyObject)
	Load	Starts a load operation. Any pending load will be implicitly canceled.
	LoadRange	If in virtual mode, loads a specific range of entities.
	ReadLocalValue	(Inherited from System.Windows.DependencyObject)
	Refresh	Starts a load operation ignoring the client-side cache. Any pending load will be implicitly canceled.
	SetValue	(Inherited from System.Windows.DependencyObject)

Top

See Also

Reference

ClientViewSource Class
C1.Data.DataSource Namespace

DeferLoad Method

Used to group changes to multiple load-affecting properties together, deferring the resulting load operations so a single load operation is performed in the end, that is, when the object returned from this method is disposed.

Syntax

Visual Basic (Declaration)
`Public Function DeferLoad() As IDisposable`
C#
`public IDisposable DeferLoad()`

Return Value

An System.IDisposable object that will trigger a Load operation when disposed using the System.IDisposable.Dispose method.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

Load Method
Starts a load operation. Any pending load will be implicitly canceled.

Syntax

Visual Basic (Declaration)
`Public Sub Load()`
C#
`public void Load()`

Remarks

If the entities are already in the cache, there will be no roundtrip to the server.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

LoadRange Method

The index of the first item to load.

The number of entities to load.

If in virtual mode, loads a specific range of entities.

Syntax

Visual Basic (Declaration)

```
Public Sub LoadRange( _
   ByVal start As Integer, _
   ByVal length As Integer _
)
```

C#

```
public void LoadRange(
   int start,
   int length
)
```

Parameters

start

The index of the first item to load.

length

The number of entities to load.

Exceptions

Exception	Description
System.InvalidOperationException	VirtualMode is VirtualModeKind.None.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

Refresh Method
Starts a load operation ignoring the client-side cache. Any pending load will be implicitly canceled.

Syntax

Visual Basic (Declaration)
`Public Sub Refresh()`
C#
`public void Refresh()`

Remarks

Use this method to refresh data with any changes that may have occurred on the server

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

Properties
For a list of all members of this type, see ClientViewSource members.

Public Properties

	Name	Description
	AutoLoad	Gets or sets a value indicating whether Load is automatically invoked on startup and when a change occurs that impacts the query composed by the ClientViewSource. The default is True.
	BaseView	Gets or sets an instance of C1.Data.ClientView<T> that the ClientViewSource uses as the base for composing queries.
	CacheTimeout	Gets or sets the period of time entities loaded in virtual mode are kept in the cache without checking whether they are needed or not. If an entity was neither used nor considered needed for a period of time longer than CacheTimeout, ClientViewSource may evict it from the cache.
	CurrentClientView	Gets the current client view used to load entities, or null in virtual mode.
	DataView	Gets the current view of entities resulting from the last load operation.
	Dispatcher	(Inherited from System.Windows.DependencyObject)
	FilterDescriptors	Gets the collection of System.Windows.Controls.FilterDescriptor objects used when performing loads.
	FilterOperator	Gets or sets the logical operator used for combinining FilterDescriptors in the filter collection. The default value is System.Windows.Controls.FilterDescriptorLogicalOperator.And.
	GroupDescriptors	Gets the collection of System.Windows.Controls.GroupDescriptor objects used to organize the loaded entities into groups.
	IsLoadingData	Gets a value indicating whether the ClientViewSource is currently loading data.
	LoadCommand	Gets an System.Windows.Input.ICommand that invokes Load on this ClientViewSource.

	LoadDelay	Gets or sets the delay before an automatic data loading operation is started. It is the delay from the time a change prompting automatic load occurs until the time the resulting Load is started. The default delay is 25 milliseconds.
	LoadSize	Gets or sets the maximum number of items to load each time a Load is executed. When equal to 0, all requested entities will be loaded. The default is 0.
	MoveToFirstOnLoad	Gets or sets a value indicating that the first item must be made current after Load operation is completed if current item was not set by other means.
	Name	Gets a name of this ClientViewSource to reference it in a C1DataSource.ViewSources collection. By default it is determined by the EntitySetName (for Entity Framework) or QueryName (for RIA Services), but it can be overrided by NameOverride.
	NameOverride	Gets or sets a value that overrides the value of the Name property.
	PageSize	Gets or sets the number of items displayed on each page of the DataView, or the number of items to fetch in each query in virtual mode, or 0 to disable paging.
	SortDescriptors	Gets the collection of System.Windows.Controls.SortDescriptor objects used to sort the data.
	VirtualMode	Gets or sets a value indicating whether the ClientViewSource is in virtual mode. Virtual mode is an innovative technology allowing to bind GUI controls directly to very large data sets without delays and performance degradation and without inconvenience of paging. By default, virtual mode is disabled (the default value is VirtualModeKind.None).

Top

See Also

Reference

ClientViewSource Class
C1.Data.DataSource Namespace

AutoLoad Property
Gets or sets a value indicating whether Load is automatically invoked on startup and when a change occurs that impacts the query composed by the ClientViewSource. The default is True.

Syntax

Visual Basic (Declaration)
`Public Property AutoLoad As Boolean`
C#
`public bool AutoLoad {get; set;}`

Remarks

When AutoLoad is True, any property change affecting the load query will automatically invoke a Load after the specified LoadDelay. Examples of properties that impact the query are PageSize and FilterOperator. Also, changes to dependency object collections like FilterDescriptors and changes to the dependency properties on elements contained in those collections will affect the query and prompt an automatic Load.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

BaseView Property
Gets or sets an instance of C1.Data.ClientView<T> that the ClientViewSource uses as the base for composing queries.

Syntax

Visual Basic (Declaration)
`Public Property BaseView As View`
C#
`public View BaseView {get; set;}`

Exceptions

Exception	Description
System.ArgumentException	The value is not null and not an instance of C1.Data.ClientView<T>.

Remarks

The ClientViewSource applies filtering, sorting, grouping, and paging to its BaseView.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

CacheTimeout Property
Gets or sets the period of time entities loaded in virtual mode are kept in the cache without checking whether they are needed or not. If an entity was neither used nor considered needed for a period of time longer than CacheTimeout, ClientViewSource may evict it from the cache.

Syntax

Visual Basic (Declaration)

```
Public Property CacheTimeout As TimeSpan
```

C#

```
public TimeSpan CacheTimeout {get; set;}
```

Remarks

This property is not used if VirtualMode is VirtualModeKind.None.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

CurrentClientView Property
Gets the current client view used to load entities, or null in virtual mode.

Syntax

Visual Basic (Declaration)

```
Public Property CurrentClientView As View
```

C#

```
public View CurrentClientView {get; set;}
```

Remarks

Using CurrentClientView, you can build client views on top of the ClientViewSource by applying live view operators to the CurrentClientView.

The value of this property changes and the PropertyChanged event is raised whenever the query used to load entities changes.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

DataView Property
Gets the current view of entities resulting from the last load operation.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property DataView As ClientCollectionView`
C#
`public ClientCollectionView DataView {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

FilterDescriptors Property
Gets the collection of System.Windows.Controls.FilterDescriptor objects used when performing loads.

Syntax

Visual Basic (Declaration)	
`Public ReadOnly Property FilterDescriptors As FilterDescriptorCollection`	
C#	
`public FilterDescriptorCollection FilterDescriptors {get;}`	

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

FilterOperator Property
Gets or sets the logical operator used for combinining FilterDescriptors in the filter collection. The default value is System.Windows.Controls.FilterDescriptorLogicalOperator.And.

Syntax

Visual Basic (Declaration)	
`Public Property FilterOperator As FilterDescriptorLogicalOperator`	
C#	
`public FilterDescriptorLogicalOperator FilterOperator {get; set;}`	

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

GroupDescriptors Property
Gets the collection of System.Windows.Controls.GroupDescriptor objects used to organize the loaded entities into groups.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property GroupDescriptors As GroupDescriptorCollection`
C#
`public GroupDescriptorCollection GroupDescriptors {get;}`

Remarks

Grouping only works in WPF and Silverlight. It is ignored in WinForms because WinForms data binding does not support grouping.

When a System.Windows.Controls.GroupDescriptor is applied, the data will inherently be sorted by the grouped property. To force a grouped property to be sorted in descending order, add a System.Windows.Controls.SortDescriptor to the SortDescriptors collection for that property using the Descending direction.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

IsLoadingData Property
Gets a value indicating whether the ClientViewSource is currently loading data.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property IsLoadingData As Boolean`
C#
`public bool IsLoadingData {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

LoadCommand Property
Gets an System.Windows.Input.ICommand that invokes Load on this ClientViewSource.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property LoadCommand As ICommand`
C#
`public ICommand LoadCommand {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

LoadDelay Property
Gets or sets the delay before an automatic data loading operation is started. It is the delay from the time a change prompting automatic load occurs until the time the resulting Load is started. The default delay is 25 milliseconds.

Syntax

Visual Basic (Declaration)
`Public Property LoadDelay As TimeSpan`
C#
`public TimeSpan LoadDelay {get; set;}`

Remarks

Multiple changes that occur within the specified time span are aggregated into a single Load operation. For every change that occurs, the delay timer is reset. This allows many changes to be combined into a single call as long as each change occurs within the specified delay from the last. Once the delay timer is allowed to elapse without a change occurring, Load will be invoked.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

LoadSize Property
Gets or sets the maximum number of items to load each time a Load is executed. When equal to 0, all requested entities will be loaded. The default is 0.

Syntax

Visual Basic (Declaration)
`Public Property LoadSize As Integer`
C#
`public int LoadSize {get; set;}`

Remarks

When PageSize and LoadSize are both non-zero, entities will be loaded using the multiple of PageSize nearest LoadSize. This allows multiple pages to be loaded at once without loading partial pages.

This property is ignored when the ClientViewSource is in virtual mode.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

MoveToFirstOnLoad Property
Gets or sets a value indicating that the first item must be made current after Load operation is completed if current item was not set by other means.

Syntax

Visual Basic (Declaration)
`Public Property MoveToFirstOnLoad As Boolean`
C#
`public bool MoveToFirstOnLoad {get; set;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

Name Property
Gets a name of this ClientViewSource to reference it in a C1DataSource.ViewSources collection. By default it is determined by the EntitySetName (for Entity Framework) or QueryName (for RIA Services), but it can be overrided by NameOverride.

Syntax

Visual Basic (Declaration)
`Public Overridable ReadOnly Property Name As String`
C#
`public virtual string Name {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

NameOverride Property
Gets or sets a value that overrides the value of the Name property.

Syntax

Visual Basic (Declaration)
`Public Property NameOverride As String`
C#
`public string NameOverride {get; set;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

PageSize Property
Gets or sets the number of items displayed on each page of the DataView, or the number of items to fetch in each query in virtual mode, or 0 to disable paging.

Syntax

Visual Basic (Declaration)
`Public Property PageSize As Integer`
C#
`public int PageSize {get; set;}`

Remarks

If not in the virtual mode, a non-zero page size will cause the number of entities loaded with each Load operation to be limited, using server-side paging.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

SortDescriptors Property
Gets the collection of System.Windows.Controls.SortDescriptor objects used to sort the data.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property SortDescriptors As SortDescriptorCollection`
C#
`public SortDescriptorCollection SortDescriptors {get;}`

Remarks

In a Load operation, the SortDescriptors are used to perform server-side sorting. The specified sorting is also applied on the client side when changes are made on the client to the loaded entities, with the DataView reflecting the changes.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

VirtualMode Property

Gets or sets a value indicating whether the ClientViewSource is in virtual mode. Virtual mode is an innovative technology allowing to bind GUI controls directly to very large data sets without delays and performance degradation and without inconvenience of paging. By default, virtual mode is disabled (the default value is VirtualModeKind.None).

Syntax

Visual Basic (Declaration)
`Public Property VirtualMode As VirtualModeKind`
C#
`public VirtualModeKind VirtualMode {get; set;}`

Remarks

ClientViewSource in virtual mode intelligently and transparently both for the end user and for the developer loads only the entities that need to be displayed on the screen.

To enable virtual mode, set this property to Managed if you have a control handler with VirtualMode set to True; otherwise, set it to VirtualModeKind.Unmanaged.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members
VirtualModeKind Enumeration

Events
>

Name	Description
LoadedData	Occurs when a load operation is completed, or when an exception was

	thrown during the load operation.
PropertyChanged	Occurs when a property value changes.

Top

See Also

Reference

ClientViewSource Class
C1.Data.DataSource Namespace

LoadedData Event
Occurs when a load operation is completed, or when an exception was thrown during the load operation.

Syntax

Visual Basic (Declaration)
`Public Event LoadedData As EventHandler(Of ClientViewLoadedEventArgs)`
C#
`public event EventHandler<ClientViewLoadedEventArgs> LoadedData`

Event Data

The event handler receives an argument of type ClientViewLoadedEventArgs containing data related to this event. The following **ClientViewLoadedEventArgs** properties provide information specific to this event.

Property	Description
Error	Gets the loading error if the loading failed.
HasError	Gets a value indicating whether the loading has failed. If true, inspect the Error property for details.
IsErrorHandled	Gets a value indicating whether the loading error has been marked as handled by calling MarkErrorAsHandled.

Items	Gets all entities loaded by a client view.
TotalItemCount	Gets the total number of rows for the original query without any paging applied to it.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSource Class
ClientViewSource Members

PropertyChanged Event
Occurs when a property value changes.

Syntax

Visual Basic (Declaration)
`Public Event PropertyChanged As PropertyChangedEventHandler`
C#
`public event PropertyChangedEventHandler PropertyChanged`

Event Data

The event handler receives an argument of type PropertyChangedEventArgs containing data related to this event. The following **PropertyChangedEventArgs** properties provide information specific to this event.

Property	Description
PropertyName	

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

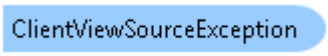
See Also

Reference

ClientViewSource Class
ClientViewSource Members

ClientViewSourceException
This exception indicates that a ClientViewSource is miconfigured or an error has occurred during the Load operation.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class ClientViewSourceException` `  Inherits System.Exception`
C#
`public class ClientViewSourceException : System.Exception`

Inheritance Hierarchy

System.Object
  System.Exception
    C1.Data.DataSource.ClientViewSourceException

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

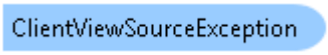
See Also

Reference

ClientViewSourceException Members
C1.Data.DataSource Namespace

Overview
This exception indicates that a ClientViewSource is miconfigured or an error has occurred during the Load operation.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class ClientViewSourceException` `  Inherits System.Exception`
C#
`public class ClientViewSourceException : System.Exception`

Inheritance Hierarchy

System.Object
  System.Exception
    C1.Data.DataSource.ClientViewSourceException

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSourceException Members
C1.Data.DataSource Namespace

Members

Properties Methods

The following tables list the members exposed by ClientViewSourceException.

Public Constructors

	Name	Description
	ClientViewSourceException Constructor	Overloaded.

Top

Public Properties

	Name	Description
	Data	(Inherited from System.Exception)
	HelpLink	(Inherited from System.Exception)
	InnerException	(Inherited from System.Exception)
	Message	(Inherited from System.Exception)
	Source	(Inherited from System.Exception)
	StackTrace	(Inherited from System.Exception)
	TargetSite	(Inherited from System.Exception)

Top

Public Methods

	Name	Description
	GetBaseException	(Inherited from System.Exception)

	GetObjectData	(Inherited from System.Exception)
	GetType	(Inherited from System.Exception)
	ToString	(Inherited from System.Exception)

Top

See Also

Reference

ClientViewSourceException Class
C1.Data.DataSource Namespace

ClientViewSourceException Constructor

Overload List

Overload	Description
ClientViewSourceException Constructor()	Initializes a new instance of the ClientViewSourceException class.
ClientViewSourceException Constructor(String)	Initializes a new instance of the ClientViewSourceException class with a specified error message.
ClientViewSourceException Constructor(String,Exception)	Initializes a new instance of the ClientViewSourceException class with a specified error message, and a reference to the inner exception that is the cause of this exception.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSourceException Class
ClientViewSourceException Members

ClientViewSourceException Constructor()
Initializes a new instance of the ClientViewSourceException class.

Syntax

Visual Basic (Declaration)	
`Public Function New()`	
C#	
`public ClientViewSourceException()`	

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSourceException Class
ClientViewSourceException Members
Overload List

ClientViewSourceException Constructor(String)
The error message that explains the reason for the exception.

Initializes a new instance of the ClientViewSourceException class with a specified error message.

Syntax

Visual Basic (Declaration)	
`Public Function New( _`	

```
   ByVal message As String _
)
```

C#

```
public ClientViewSourceException(
   string message
)
```

Parameters

message

The error message that explains the reason for the exception.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSourceException Class
ClientViewSourceException Members
Overload List

ClientViewSourceException Constructor(String,Exception)
The error message that explains the reason for the exception.

The exception that is the cause of the current exception.

Initializes a new instance of the ClientViewSourceException class with a specified error message, and a reference to the inner exception that is the cause of this exception.

Syntax

Visual Basic (Declaration)

```
Public Function New( _
   ByVal message As String, _
   ByVal inner As Exception _
```

```
)
```

C#

```
public ClientViewSourceException(
   string message,
   Exception inner
)
```

Parameters

message

The error message that explains the reason for the exception.

inner

The exception that is the cause of the current exception.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientViewSourceException Class
ClientViewSourceException Members
Overload List

Enumerations

VirtualModeKind
Enumeration of possible virtual modes a ClientViewSource can be in. Used in the VirtualMode property.

Syntax

Visual Basic (Declaration)

```
Public Enum VirtualModeKind
   Inherits System.Enum
```

C#
`public enum VirtualModeKind : System.Enum`

Members

Member	Description
Managed	Virtual mode is managed by a GUI control bound to the ClientViewSource. That GUI control must have a control handler with the VirtualMode property set to True.
None	Virtual mode is disabled.
Unmanaged	Virtual mode is not managed by a control handler, it is managed by the ClientViewSource itself that is unaware of what controls are bound to it. This option should be used only if you don't have a GUI control that supports virtual mode through a control handler. Although it allows to use virtual mode with any GUI bound control (with or without a control handler), it should be used with caution, only if you can't use the Managed option. See the "Programming Guide" in the Studio for Entity Framework documentation for more details.

Inheritance Hierarchy

System.Object
  System.ValueType
    System.Enum
      C1.Data.DataSource.VirtualModeKind

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1.Data.DataSource Namespace

C1.Data.Transactions Namespace

Overview

Classes

	Class	Description
	ClientTransaction	Represents a transaction that tracks client-side changes and can roll them back.

See Also

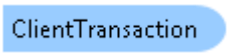
Reference

C1.Silverlight.Data.Entity Assembly

Classes

ClientTransaction
Represents a transaction that tracks client-side changes and can roll them back.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class ClientTransaction` `  Implements C1.LiveLinq.ITransaction`
C#
`public class ClientTransaction : C1.LiveLinq.ITransaction`

Remarks

To create a new independent transaction, use the C1.Data.ClientCacheBase.CreateTransaction method. To create a child transaction, use the constructor.

Inheritance Hierarchy

System.Object
C1.Data.Transactions.ClientTransaction

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

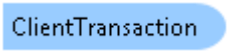
See Also

Reference

ClientTransaction Members
C1.Data.Transactions Namespace

Overview
Represents a transaction that tracks client-side changes and can roll them back.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class ClientTransaction` `  Implements C1.LiveLinq.ITransaction`
C#
`public class ClientTransaction : C1.LiveLinq.ITransaction`

Remarks

To create a new independent transaction, use the C1.Data.ClientCacheBase.CreateTransaction method. To create a child transaction, use the constructor.

Inheritance Hierarchy

System.Object
C1.Data.Transactions.ClientTransaction

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientTransaction Members
C1.Data.Transactions Namespace

Members
Properties Methods Events

The following tables list the members exposed by ClientTransaction.

Public Constructors

	Name	Description
	ClientTransaction Constructor	Initializes a child (nested) transaction, a new instance of the ClientTransaction class with a specified parent transaction.

Top

Public Properties

	Name	Description
	HasChanges	Gets a value indicating whether any changes were made in the scope of this transaction.
	State	Gets the state the transaction is in.

Top

Public Methods

	Name	Description

	Commit	Commits the transaction if it was not committed before. Commits changes that were made while this transaction's scope was open.
	Dispose	Disposes of the ClientTransaction.
	Rollback	Rolls back the transaction.
	Scope	Opens the transaction scope.
	ScopeDataContext	Wraps an object so the transaction scope is automatically opened when a value is being assigned to a property of the wrapped object.

Top

Public Events

	Name	Description
	PropertyChanged	Occurs when a property value changes, after it has been changed.

Top

See Also

Reference

ClientTransaction Class
C1.Data.Transactions Namespace

ClientTransaction Constructor
The parent transaction. Cannot be null.

Initializes a child (nested) transaction, a new instance of the ClientTransaction class with a specified parent transaction.

Syntax

Visual Basic (Declaration)	
`Public Function New( _`	

```
   ByVal parent As ClientTransaction _
)
```

C#

```
public ClientTransaction(
   ClientTransaction parent
)
```

Parameters

parent

The parent transaction. Cannot be null.

Exceptions

Exception	Description
System.ArgumentNullException	The parent is null.

Remarks

A child transaction is automatically committed/rolled back if its parent transaction is committed/rolled back.

Create a child transaction in cases where you need to open a new window for editing a portion of data that is being editing in an already open transaction.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientTransaction Class
ClientTransaction Members

Methods

For a list of all members of this type, see ClientTransaction members.

Public Methods

	Name	Description
	Commit	Commits the transaction if it was not committed before. Commits changes that were made while this transaction's scope was open.
	Dispose	Disposes of the ClientTransaction.
	Rollback	Rolls back the transaction.
	Scope	Opens the transaction scope.
	ScopeDataContext	Wraps an object so the transaction scope is automatically opened when a value is being assigned to a property of the wrapped object.

Top

See Also

Reference

ClientTransaction Class
C1.Data.Transactions Namespace

Commit Method
Commits the transaction if it was not committed before. Commits changes that were made while this transaction's scope was open.

Syntax

Visual Basic (Declaration)

```
Public Sub Commit()
```

C#

```
public void Commit()
```

Exceptions

Exception	Description
System.InvalidOperationException	The State is not C1.LiveLinq.TransactionState.Open.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientTransaction Class
ClientTransaction Members

Dispose Method
Disposes of the ClientTransaction.

Syntax

Visual Basic (Declaration)
`Public Sub Dispose()`
C#
`public void Dispose()`

Remarks

If the State is C1.LiveLinq.TransactionState.Open, the ClientTransaction is automatically rolled back.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientTransaction Class
ClientTransaction Members

Rollback Method
Rolls back the transaction.

Syntax

Visual Basic (Declaration)
`Public Sub Rollback()`
C#
`public void Rollback()`

Exceptions

Exception	Description
System.InvalidOperationException	The State is C1.LiveLinq.TransactionState.Committed or C1.LiveLinq.TransactionState.Committing.

Remarks

Calling this method cancels the changes that were made in the scope of this transaction.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientTransaction Class
ClientTransaction Members

Scope Method
Opens the transaction scope.

Syntax

Visual Basic (Declaration)
`Public Function Scope() As IDisposable`
C#
`public IDisposable Scope()`

Return Value

An instance of an System.IDisposable that will close the scope when its System.IDisposable.Dispose method is called.

Exceptions

Exception	Description
System.InvalidOperationException	The **ClientTransaction.State** is not C1.LiveLinq.TransactionState.Open.

Remarks

The transaction tracks changes only when they are made inside an open scope.

Calling System.IDisposable.Dispose on the return value closes the scope.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientTransaction Class
ClientTransaction Members

ScopeDataContext Method
The object to wrap.

Wraps an object so the transaction scope is automatically opened when a value is being assigned to a property of the wrapped object.

Syntax

Visual Basic (Declaration)

```
Public Function ScopeDataContext( _
   ByVal entity As Object _
) As Object
```

C#

```
public object ScopeDataContext(
   object entity
)
```

Parameters

entity

The object to wrap.

Return Value

The wrapped object.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientTransaction Class
ClientTransaction Members

Properties
For a list of all members of this type, see ClientTransaction members.

Public Properties

	Name	Description
	HasChanges	Gets a value indicating whether any changes were made in the scope of this transaction.
	State	Gets the state the transaction is in.

Top

See Also

Reference

ClientTransaction Class
C1.Data.Transactions Namespace

HasChanges Property
Gets a value indicating whether any changes were made in the scope of this transaction.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property HasChanges As Boolean`
C#
`public bool HasChanges {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientTransaction Class
ClientTransaction Members

State Property
Gets the state the transaction is in.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property State As TransactionState`
C#
`public TransactionState State {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientTransaction Class
ClientTransaction Members
TransactionState Enumeration

Events
For a list of all members of this type, see ClientTransaction members.

Public Events

	Name	Description
	PropertyChanged	Occurs when a property value changes, after it has been changed.

Top

See Also

Reference

ClientTransaction Class
C1.Data.Transactions Namespace

PropertyChanged Event
Occurs when a property value changes, after it has been changed.

Syntax

Visual Basic (Declaration)	
`Public Event PropertyChanged As PropertyChangedEventHandler`	
C#	
`public event PropertyChangedEventHandler PropertyChanged`	

Event Data

The event handler receives an argument of type PropertyChangedEventArgs containing data related to this event. The following **PropertyChangedEventArgs** properties provide information specific to this event.

Property	Description
PropertyName	

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ClientTransaction Class
ClientTransaction Members

C1.Silverlight.Data Namespace

Overview

Classes

	Class	Description
	ControlHandler	Represents a control handler that connect GUI controls of supported types to a C1DataSource so that those controls can be given additional

		functionality such as lookup columns and virtual mode.

See Also

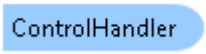
Reference

C1.Silverlight.Data.Entity Assembly

Classes

ControlHandler
Represents a control handler that connect GUI controls of supported types to a C1DataSource so that those controls can be given additional functionality such as lookup columns and virtual mode.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class ControlHandler` `  Inherits C1.Data.DataSource.BaseControlHandler`
C#
`public class ControlHandler : C1.Data.DataSource.BaseControlHandler`

Remarks

To connect a control handler to a GUI control, assign an instance of this class to the C1.Silverlight.Data.RiaServices.C1DataSource.ControlHandlerProperty attached property of the GUI control.

The supported GUI controls are: System.Windows.Controls.DataGrid, C1.Silverlight.FlexGrid.C1FlexGrid, C1.Silverlight.DataGrid.C1DataGrid.

Lookup columns are not supported in System.Windows.Controls.DataGrid because that control lacks required functionality.

Inheritance Hierarchy

System.Object
  System.Windows.DependencyObject
    C1.Data.DataSource.BaseControlHandler
      C1.Silverlight.Data.ControlHandler

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

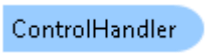
See Also

Reference

ControlHandler Members
C1.Silverlight.Data Namespace

Overview
Represents a control handler that connect GUI controls of supported types to a C1DataSource so that those controls can be given additional functionality such as lookup columns and virtual mode.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class ControlHandler` `  Inherits C1.Data.DataSource.BaseControlHandler`
C#
`public class ControlHandler : C1.Data.DataSource.BaseControlHandler`

Remarks

To connect a control handler to a GUI control, assign an instance of this class to the C1.Silverlight.Data.RiaServices.C1DataSource.ControlHandlerProperty attached property of the GUI control.

The supported GUI controls are: System.Windows.Controls.DataGrid, C1.Silverlight.FlexGrid.C1FlexGrid, C1.Silverlight.DataGrid.C1DataGrid.

Lookup columns are not supported in System.Windows.Controls.DataGrid because that control lacks required functionality.

Inheritance Hierarchy

System.Object
  System.Windows.DependencyObject
    C1.Data.DataSource.BaseControlHandler
      C1.Silverlight.Data.ControlHandler

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ControlHandler Members
C1.Silverlight.Data Namespace

Members
Fields Properties Methods

The following tables list the members exposed by ControlHandler.

Public Constructors

	Name	Description
	ControlHandler Constructor	

Top

Public Fields

	Name	Description

S	DataSourceProperty	The DependencyProperty for the DataSource property.

Top

Public Properties

	Name	Description
	AutoLookup	Gets or sets a value indicating whether data grid columns bound to navigation (foreign key, lookup) properties must be converted to combo box columns, so the user can see the right value and edit it by choosing a value from a drop-down list. The default value is False. (Inherited from C1.Data.DataSource.BaseControlHandler)
	DataSource	Gets or sets a data source of the control handler.
	Dispatcher	(Inherited from System.Windows.DependencyObject)
	SupportsVirtualMode	Gets a value indicating whether this control handler supports Virtual Mode. (Inherited from C1.Data.DataSource.BaseControlHandler)
	VirtualMode	Gets or sets a value indicating whether virtual mode specified in a C1.Data.DataSource.ClientViewSource is managed by this control handler. (Inherited from C1.Data.DataSource.BaseControlHandler)

Top

Public Methods

	Name	Description
	Apply	Forces this control handler to apply its settings to the current control. (Inherited from C1.Data.DataSource.BaseControlHandler)
	ClearValue	(Inherited from System.Windows.DependencyObject)

	GetAnimationBaseValue	(Inherited from System.Windows.DependencyObject)
	GetValue	(Inherited from System.Windows.DependencyObject)
	ReadLocalValue	(Inherited from System.Windows.DependencyObject)
	SetValue	(Inherited from System.Windows.DependencyObject)

Top

See Also

Reference

ControlHandler Class
C1.Silverlight.Data Namespace

ControlHandler Constructor

Syntax

Visual Basic (Declaration)
`Public Function New()`
C#
`public ControlHandler()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ControlHandler Class
ControlHandler Members

Properties

For a list of all members of this type, see ControlHandler members.

Public Properties

	Name	Description
	AutoLookup	Gets or sets a value indicating whether data grid columns bound to navigation (foreign key, lookup) properties must be converted to combo box columns, so the user can see the right value and edit it by choosing a value from a drop-down list. The default value is False. (Inherited from C1.Data.DataSource.BaseControlHandler)
	DataSource	Gets or sets a data source of the control handler.
	Dispatcher	(Inherited from System.Windows.DependencyObject)
	SupportsVirtualMode	Gets a value indicating whether this control handler supports Virtual Mode. (Inherited from C1.Data.DataSource.BaseControlHandler)
	VirtualMode	Gets or sets a value indicating whether virtual mode specified in a C1.Data.DataSource.ClientViewSource is managed by this control handler. (Inherited from C1.Data.DataSource.BaseControlHandler)

Top

See Also

Reference

ControlHandler Class
C1.Silverlight.Data Namespace

DataSource Property
Gets or sets a data source of the control handler.

Syntax

Visual Basic (Declaration)

```
Public Property DataSource As C1DataSource
```

C#	
`public C1DataSource DataSource {get; set;}`	

Remarks

Setting this property is required if the GUI control is not bound directly to a C1DataSource. For example, it must be set if the GUI control is bound to a live view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ControlHandler Class
ControlHandler Members

Fields
For a list of all members of this type, see ControlHandler members.

Public Fields

	Name	Description
S	DataSourceProperty	The DependencyProperty for the DataSource property.

Top

See Also

Reference

ControlHandler Class
C1.Silverlight.Data Namespace

DataSourceProperty Field
The DependencyProperty for the DataSource property.

Syntax

Visual Basic (Declaration)
`Public Shared ReadOnly DataSourceProperty As DependencyProperty`
C#
`public static readonly DependencyProperty DataSourceProperty`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ControlHandler Class
ControlHandler Members

C1.Silverlight.Data.RiaServices Namespace

Overview

Classes

	Class	Description
	C1DataSource	A data source control that simplifies data binding of GUI controls to data in a RiaClientCache. Can be used to bind multiple controls to different queries.
	RiaClientCache	Represents a client-side cache specific to RIA Services.
	RiaClientScope	Defines a scope of data access. Provides facilities to create client views.
	RiaServicesExtensions	Provides a set of extensions methods for RIA Services.
	RiaViewSource	A RIA Services-specific version of the

		C1.Data.DataSource.ClientViewSource class.
	RiaViewSourceCollection	An observable collection of RiaViewSource objects.

See Also

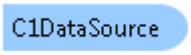
Reference

C1.Silverlight.Data.Entity Assembly

Classes

C1DataSource
A data source control that simplifies data binding of GUI controls to data in a RiaClientCache. Can be used to bind multiple controls to different queries.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class C1DataSource` `  Inherits System.Windows.Controls.Control`
C#
`public class C1DataSource : System.Windows.Controls.Control`

Remarks

To bind a control to data in a RiaClientCache, add a C1DataSource to a XAML file, specify the context type, populate the ViewSources collection with RiaViewSource objects to define views (based on queries), and bind GUI controls like this: `<DataGrid ItemsSource="{Binding Customers, ElementName=c1DataSource}"/>` where Customers is the name of a RiaViewSource in the ViewSources collection, and c1DataSource is the name of the C1DataSource.

Inheritance Hierarchy

System.Object
  System.Windows.DependencyObject

System.Windows.UIElement
System.Windows.FrameworkElement
System.Windows.Controls.Control
C1.Silverlight.Data.RiaServices.C1DataSource

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

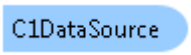
See Also

Reference

C1DataSource Members
C1.Silverlight.Data.RiaServices Namespace

Overview
A data source control that simplifies data binding of GUI controls to data in a RiaClientCache. Can be used to bind multiple controls to different queries.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class C1DataSource` `  Inherits System.Windows.Controls.Control`
C#
`public class C1DataSource : System.Windows.Controls.Control`

Remarks

To bind a control to data in a RiaClientCache, add a C1DataSource to a XAML file, specify the context type, populate the ViewSources collection with RiaViewSource objects to define views (based on queries), and bind GUI controls like this: `<DataGrid ItemsSource="{Binding`

`Customers, ElementName=c1DataSource}"/>` where Customers is the name of a RiaViewSource in the ViewSources collection, and c1DataSource is the name of the C1DataSource.

Inheritance Hierarchy

System.Object
  System.Windows.DependencyObject
    System.Windows.UIElement
      System.Windows.FrameworkElement
        System.Windows.Controls.Control
          C1.Silverlight.Data.RiaServices.C1DataSource

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Members
C1.Silverlight.Data.RiaServices Namespace

Members
Fields Properties Methods Events

The following tables list the members exposed by C1DataSource.

Public Constructors

	Name	Description
	C1DataSource Constructor	Initializes a new instance of the C1DataSource class.

Top

Public Fields

	Name	Description

S	ControlHandlerProperty	Identifies the C1.Silverlight.Data.RiaServices.C1DataSource.ControlHandler attached property.

Top

Public Properties

	Name	Description
	ActualHeight	(Inherited from System.Windows.FrameworkElement)
	ActualWidth	(Inherited from System.Windows.FrameworkElement)
	AllowDrop	(Inherited from System.Windows.UIElement)
	Background	(Inherited from System.Windows.Controls.Control)
	BorderBrush	(Inherited from System.Windows.Controls.Control)
	BorderThickness	(Inherited from System.Windows.Controls.Control)
	CacheMode	(Inherited from System.Windows.UIElement)
	ClientCache	Gets or sets the RiaClientCache used by this C1DataSource to access the data.
	ClientScope	Gets the client scope to which this C1DataSource belongs.
	Clip	(Inherited from System.Windows.UIElement)
	Cursor	(Inherited from System.Windows.FrameworkElement)
	DataContext	(Inherited from System.Windows.FrameworkElement)
	DesiredSize	(Inherited from System.Windows.UIElement)

	Dispatcher	(Inherited from System.Windows.DependencyObject)
	DomainContext	Gets the System.ServiceModel.DomainServices.Client.DomainContext the ClientCache is connected to.
	DomainContextTypeName	Gets or sets the full name of a System.ServiceModel.DomainServices.Client.DomainContext type used to obtain the default client cache.
	Effect	(Inherited from System.Windows.UIElement)
	FlowDirection	(Inherited from System.Windows.FrameworkElement)
	FontFamily	(Inherited from System.Windows.Controls.Control)
	FontSize	(Inherited from System.Windows.Controls.Control)
	FontStretch	(Inherited from System.Windows.Controls.Control)
	FontStyle	(Inherited from System.Windows.Controls.Control)
	FontWeight	(Inherited from System.Windows.Controls.Control)
	Foreground	(Inherited from System.Windows.Controls.Control)
	Height	(Inherited from System.Windows.FrameworkElement)
	HorizontalAlignment	(Inherited from System.Windows.FrameworkElement)
	HorizontalContentAlignment	(Inherited from System.Windows.Controls.Control)
	IsEnabled	(Inherited from System.Windows.Controls.Control)
	IsHitTestVisible	(Inherited from System.Windows.UIElement)

	IsTabStop	(Inherited from System.Windows.Controls.Control)
	Item	Overloaded. Gets the C1.Data.DataSource.ClientCollectionView of the RiaViewSource with the specified name in the ViewSources collection.
	Language	(Inherited from System.Windows.FrameworkElement)
	Margin	(Inherited from System.Windows.FrameworkElement)
	MaxHeight	(Inherited from System.Windows.FrameworkElement)
	MaxWidth	(Inherited from System.Windows.FrameworkElement)
	MinHeight	(Inherited from System.Windows.FrameworkElement)
	MinWidth	(Inherited from System.Windows.FrameworkElement)
	Name	(Inherited from System.Windows.FrameworkElement)
	Opacity	(Inherited from System.Windows.UIElement)
	OpacityMask	(Inherited from System.Windows.UIElement)
	Padding	(Inherited from System.Windows.Controls.Control)
	Parent	(Inherited from System.Windows.FrameworkElement)
	Projection	(Inherited from System.Windows.UIElement)
	RefreshInterval	Gets or sets the interval between automatic Refresh operations to refresh the data with any changes that may have occurred on the server.
	RenderSize	(Inherited from System.Windows.UIElement)

	RenderTransform	(Inherited from System.Windows.UIElement)
	RenderTransformOrigin	(Inherited from System.Windows.UIElement)
	Resources	(Inherited from System.Windows.FrameworkElement)
	Style	(Inherited from System.Windows.FrameworkElement)
	TabIndex	(Inherited from System.Windows.Controls.Control)
	TabNavigation	(Inherited from System.Windows.Controls.Control)
	Tag	(Inherited from System.Windows.FrameworkElement)
	Template	(Inherited from System.Windows.Controls.Control)
	Triggers	(Inherited from System.Windows.FrameworkElement)
	UseLayoutRounding	(Inherited from System.Windows.UIElement)
	VerticalAlignment	(Inherited from System.Windows.FrameworkElement)
	VerticalContentAlignment	(Inherited from System.Windows.Controls.Control)
	ViewSources	Gets a collection of RiaViewSource objects that define views (based on queries) in this C1DataSource.
	Visibility	(Inherited from System.Windows.UIElement)
	Width	(Inherited from System.Windows.FrameworkElement)

Top

Public Methods

	Name	Description

	AddHandler	(Inherited from System.Windows.UIElement)
	ApplyTemplate	(Inherited from System.Windows.Controls.Control)
	Arrange	(Inherited from System.Windows.UIElement)
	CaptureMouse	(Inherited from System.Windows.UIElement)
	ClearValue	(Inherited from System.Windows.DependencyObject)
	FindName	(Inherited from System.Windows.FrameworkElement)
	Focus	(Inherited from System.Windows.Controls.Control)
	GetAnimationBaseValue	(Inherited from System.Windows.DependencyObject)
	GetBindingExpression	(Inherited from System.Windows.FrameworkElement)
S	GetControlHandler	Gets the value of the C1.Silverlight.Data.RiaServices.C1DataSource.ControlHandler attached property from a given control.
	GetValue	(Inherited from System.Windows.DependencyObject)
	InvalidateArrange	(Inherited from System.Windows.UIElement)
	InvalidateMeasure	(Inherited from System.Windows.UIElement)
	Load	Loads all RiaViewSource objects in the ViewSources collection.
	Measure	(Inherited from System.Windows.UIElement)
	OnApplyTemplate	(Inherited from System.Windows.FrameworkElement)

	ReadLocalValue	(Inherited from System.Windows.DependencyObject)
	Refresh	Refreshes all RiaViewSource objects in the ViewSources collection.
	RejectChanges	Rejects the changes for every entity in the DomainContext.
	ReleaseMouseCapture	(Inherited from System.Windows.UIElement)
	RemoveHandler	(Inherited from System.Windows.UIElement)
	SaveChanges	Sumbits all changes to the server.
	SetBinding	(Inherited from System.Windows.FrameworkElement)
S	SetControlHandler	Sets the value of the C1.Silverlight.Data.RiaServices.C1DataSource.ControlHandler attached property to a given control.
	SetValue	(Inherited from System.Windows.DependencyObject)
	TransformToVisual	(Inherited from System.Windows.UIElement)
	UpdateLayout	(Inherited from System.Windows.UIElement)

Top

Public Events

	Name	Description
	BindingValidationError	(Inherited from System.Windows.FrameworkElement)
	DragEnter	(Inherited from System.Windows.UIElement)
	DragLeave	(Inherited from System.Windows.UIElement)

	DragOver	(Inherited from System.Windows.UIElement)
	Drop	(Inherited from System.Windows.UIElement)
	GotFocus	(Inherited from System.Windows.UIElement)
	IsEnabledChanged	(Inherited from System.Windows.Controls.Control)
	KeyDown	(Inherited from System.Windows.UIElement)
	KeyUp	(Inherited from System.Windows.UIElement)
	LayoutUpdated	(Inherited from System.Windows.FrameworkElement)
	Loaded	(Inherited from System.Windows.FrameworkElement)
	LostFocus	(Inherited from System.Windows.UIElement)
	LostMouseCapture	(Inherited from System.Windows.UIElement)
	MouseEnter	(Inherited from System.Windows.UIElement)
	MouseLeave	(Inherited from System.Windows.UIElement)
	MouseLeftButtonDown	(Inherited from System.Windows.UIElement)
	MouseLeftButtonUp	(Inherited from System.Windows.UIElement)
	MouseMove	(Inherited from System.Windows.UIElement)
	MouseRightButtonDown	(Inherited from System.Windows.UIElement)
	MouseRightButtonUp	(Inherited from System.Windows.UIElement)
	MouseWheel	(Inherited from System.Windows.UIElement)

	SavedChanges	Occurs after a submit operation is completed.
	SavingChanges	Occurs before changes are submitted.
	SizeChanged	(Inherited from System.Windows.FrameworkElement)
	TextInput	(Inherited from System.Windows.UIElement)
	TextInputStart	(Inherited from System.Windows.UIElement)
	TextInputUpdate	(Inherited from System.Windows.UIElement)
	Unloaded	(Inherited from System.Windows.FrameworkElement)

Top

See Also

Reference

C1DataSource Class
C1.Silverlight.Data.RiaServices Namespace

C1DataSource Constructor
Initializes a new instance of the C1DataSource class.

Syntax

Visual Basic (Declaration)
`Public Function New()`
C#
`public C1DataSource()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

Methods
For a list of all members of this type, see C1DataSource members.

Public Methods

	Name	Description
	AddHandler	(Inherited from System.Windows.UIElement)
	ApplyTemplate	(Inherited from System.Windows.Controls.Control)
	Arrange	(Inherited from System.Windows.UIElement)
	CaptureMouse	(Inherited from System.Windows.UIElement)
	ClearValue	(Inherited from System.Windows.DependencyObject)
	FindName	(Inherited from System.Windows.FrameworkElement)
	Focus	(Inherited from System.Windows.Controls.Control)
	GetAnimationBaseValue	(Inherited from System.Windows.DependencyObject)
	GetBindingExpression	(Inherited from System.Windows.FrameworkElement)
S	GetControlHandler	Gets the value of the C1.Silverlight.Data.RiaServices.C1DataSource.ControlHandler attached property from a given control.

	GetValue	(Inherited from System.Windows.DependencyObject)
	InvalidateArrange	(Inherited from System.Windows.UIElement)
	InvalidateMeasure	(Inherited from System.Windows.UIElement)
	Load	Loads all RiaViewSource objects in the ViewSources collection.
	Measure	(Inherited from System.Windows.UIElement)
	OnApplyTemplate	(Inherited from System.Windows.FrameworkElement)
	ReadLocalValue	(Inherited from System.Windows.DependencyObject)
	Refresh	Refreshes all RiaViewSource objects in the ViewSources collection.
	RejectChanges	Rejects the changes for every entity in the DomainContext.
	ReleaseMouseCapture	(Inherited from System.Windows.UIElement)
	RemoveHandler	(Inherited from System.Windows.UIElement)
	SaveChanges	Sumbits all changes to the server.
	SetBinding	(Inherited from System.Windows.FrameworkElement)
S	SetControlHandler	Sets the value of the C1.Silverlight.Data.RiaServices.C1DataSource.ControlHandler attached property to a given control.
	SetValue	(Inherited from System.Windows.DependencyObject)
	TransformToVisual	(Inherited from System.Windows.UIElement)
	UpdateLayout	(Inherited from System.Windows.UIElement)

Top

See Also

Reference

C1DataSource Class
C1.Silverlight.Data.RiaServices Namespace

GetControlHandler Method
The control from which to get the property value.

Gets the value of the C1.Silverlight.Data.RiaServices.C1DataSource.ControlHandler attached property from a given control.

Syntax

Visual Basic (Declaration)

```
Public Shared Function GetControlHandler( _
   ByVal control As DependencyObject _
) As BaseControlHandler
```

C#

```
public static BaseControlHandler GetControlHandler(
   DependencyObject control
)
```

Parameters

control

The control from which to get the property value.

Return Value

The value of the C1.Silverlight.Data.RiaServices.C1DataSource.ControlHandler attached property.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members
ControlHandlerProperty Field

Load Method
Loads all RiaViewSource objects in the ViewSources collection.

Syntax

Visual Basic (Declaration)
`Public Sub Load()`
C#
`public void Load()`

Remarks

This method calls Load for all elements of the ViewSources collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

Refresh Method
Refreshes all RiaViewSource objects in the ViewSources collection.

Syntax

Visual Basic (Declaration)
`Public Sub Refresh()`
C#
`public void Refresh()`

Remarks

This method calls C1.Data.DataSource.ClientViewSource.Refresh for all elements of the ViewSources collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

RejectChanges Method
Rejects the changes for every entity in the DomainContext.

Syntax

Visual Basic (Declaration)
`Public Sub RejectChanges()`
C#
`public void RejectChanges()`

Remarks

Changes will be rejected for all entities in the DomainContext, including those that were not loaded through this C1DataSource. This will also cancel a pending Add or Edit in the collection views.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

SaveChanges Method
Sumbits all changes to the server.

Syntax

Visual Basic (Declaration)	
`Public Sub SaveChanges()`	
C#	
`public void SaveChanges()`	

Remarks

Changes will be submitted for all modified/added/deleted entities in the DomainContext, including those that were not loaded through this C1DataSource. This will also commit a pending Add or Edit in the collection views.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

SetControlHandler Method

The object on which to set the C1.Silverlight.Data.RiaServices.C1DataSource.ControlHandler attached property.

The property value to set.

Sets the value of the C1.Silverlight.Data.RiaServices.C1DataSource.ControlHandler attached property to a given control.

Syntax

Visual Basic (Declaration)

```
Public Shared Sub SetControlHandler( _
   ByVal control As DependencyObject, _
   ByVal handler As BaseControlHandler _
)
```

C#

```
public static void SetControlHandler(
   DependencyObject control,
   BaseControlHandler handler
)
```

Parameters

control

The object on which to set the C1.Silverlight.Data.RiaServices.C1DataSource.ControlHandler attached property.

handler

The property value to set.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members
ControlHandlerProperty Field

Properties
For a list of all members of this type, see C1DataSource members.

Public Properties

	Name	Description
	ActualHeight	(Inherited from System.Windows.FrameworkElement)
	ActualWidth	(Inherited from System.Windows.FrameworkElement)
	AllowDrop	(Inherited from System.Windows.UIElement)
	Background	(Inherited from System.Windows.Controls.Control)
	BorderBrush	(Inherited from System.Windows.Controls.Control)
	BorderThickness	(Inherited from System.Windows.Controls.Control)
	CacheMode	(Inherited from System.Windows.UIElement)
	ClientCache	Gets or sets the RiaClientCache used by this C1DataSource to access the data.
	ClientScope	Gets the client scope to which this C1DataSource belongs.
	Clip	(Inherited from System.Windows.UIElement)
	Cursor	(Inherited from System.Windows.FrameworkElement)
	DataContext	(Inherited from System.Windows.FrameworkElement)
	DesiredSize	(Inherited from System.Windows.UIElement)

	Dispatcher	(Inherited from System.Windows.DependencyObject)
	DomainContext	Gets the System.ServiceModel.DomainServices.Client.DomainContext the ClientCache is connected to.
	DomainContextTypeName	Gets or sets the full name of a System.ServiceModel.DomainServices.Client.DomainContext type used to obtain the default client cache.
	Effect	(Inherited from System.Windows.UIElement)
	FlowDirection	(Inherited from System.Windows.FrameworkElement)
	FontFamily	(Inherited from System.Windows.Controls.Control)
	FontSize	(Inherited from System.Windows.Controls.Control)
	FontStretch	(Inherited from System.Windows.Controls.Control)
	FontStyle	(Inherited from System.Windows.Controls.Control)
	FontWeight	(Inherited from System.Windows.Controls.Control)
	Foreground	(Inherited from System.Windows.Controls.Control)
	Height	(Inherited from System.Windows.FrameworkElement)
	HorizontalAlignment	(Inherited from System.Windows.FrameworkElement)
	HorizontalContentAlignment	(Inherited from System.Windows.Controls.Control)
	IsEnabled	(Inherited from System.Windows.Controls.Control)
	IsHitTestVisible	(Inherited from System.Windows.UIElement)

	IsTabStop	(Inherited from System.Windows.Controls.Control)
	Item	Overloaded. Gets the C1.Data.DataSource.ClientCollectionView of the RiaViewSource with the specified name in the ViewSources collection.
	Language	(Inherited from System.Windows.FrameworkElement)
	Margin	(Inherited from System.Windows.FrameworkElement)
	MaxHeight	(Inherited from System.Windows.FrameworkElement)
	MaxWidth	(Inherited from System.Windows.FrameworkElement)
	MinHeight	(Inherited from System.Windows.FrameworkElement)
	MinWidth	(Inherited from System.Windows.FrameworkElement)
	Name	(Inherited from System.Windows.FrameworkElement)
	Opacity	(Inherited from System.Windows.UIElement)
	OpacityMask	(Inherited from System.Windows.UIElement)
	Padding	(Inherited from System.Windows.Controls.Control)
	Parent	(Inherited from System.Windows.FrameworkElement)
	Projection	(Inherited from System.Windows.UIElement)
	RefreshInterval	Gets or sets the interval between automatic Refresh operations to refresh the data with any changes that may have occurred on the server.
	RenderSize	(Inherited from System.Windows.UIElement)

	RenderTransform	(Inherited from System.Windows.UIElement)
	RenderTransformOrigin	(Inherited from System.Windows.UIElement)
	Resources	(Inherited from System.Windows.FrameworkElement)
	Style	(Inherited from System.Windows.FrameworkElement)
	TabIndex	(Inherited from System.Windows.Controls.Control)
	TabNavigation	(Inherited from System.Windows.Controls.Control)
	Tag	(Inherited from System.Windows.FrameworkElement)
	Template	(Inherited from System.Windows.Controls.Control)
	Triggers	(Inherited from System.Windows.FrameworkElement)
	UseLayoutRounding	(Inherited from System.Windows.UIElement)
	VerticalAlignment	(Inherited from System.Windows.FrameworkElement)
	VerticalContentAlignment	(Inherited from System.Windows.Controls.Control)
	ViewSources	Gets a collection of RiaViewSource objects that define views (based on queries) in this C1DataSource.
	Visibility	(Inherited from System.Windows.UIElement)
	Width	(Inherited from System.Windows.FrameworkElement)

Top

See Also

Reference

C1DataSource Class
C1.Silverlight.Data.RiaServices Namespace

ClientCache Property
Gets or sets the RiaClientCache used by this C1DataSource to access the data.

Syntax

Visual Basic (Declaration)
`Public Property ClientCache As RiaClientCache`
C#
`public RiaClientCache ClientCache {get; set;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

ClientScope Property
Gets the client scope to which this C1DataSource belongs.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property ClientScope As RiaClientScope`
C#
`public RiaClientScope ClientScope {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

DomainContext Property
Gets the System.ServiceModel.DomainServices.Client.DomainContext the ClientCache is connected to.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property DomainContext As DomainContext`
C#
`public DomainContext DomainContext {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

DomainContextTypeName Property
Gets or sets the full name of a System.ServiceModel.DomainServices.Client.DomainContext type used to obtain the default client cache.

Syntax

Visual Basic (Declaration)
`Public Property DomainContextTypeName As String`
C#
`public string DomainContextTypeName {get; set;}`

Remarks

The property value must satisfy the requirements for the 'typeName' parameter of the System.Type.GetType(System.String) method.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

Item Property
Gets the C1.Data.DataSource.ClientCollectionView of the RiaViewSource with the specified name in the ViewSources collection.

Overload List

Overload	Description
Item(String)	Gets the C1.Data.DataSource.ClientCollectionView of the RiaViewSource with the specified name in the ViewSources collection.
Item(Int32)	Gets the C1.Data.DataSource.ClientCollectionView of the RiaViewSource at the specified index in the ViewSources collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

Item(String) Property
The name of the RiaViewSource to take the C1.Data.DataSource.ClientCollectionView from.

Gets the C1.Data.DataSource.ClientCollectionView of the RiaViewSource with the specified name in the ViewSources collection.

Syntax

Visual Basic (Declaration)

`Public Overloads ReadOnly Property Item( _`
`  ByVal name As String _`
`) As ClientCollectionView`

C#

`public ClientCollectionView Item(`
`  string name`
`) {get;}`

Parameters

name

The name of the RiaViewSource to take the C1.Data.DataSource.ClientCollectionView from.

Property Value

The C1.Data.DataSource.ClientCollectionView of the RiaViewSource.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members
Overload List

Item(Int32) Property
The index of the RiaViewSource to take the C1.Data.DataSource.ClientCollectionView from.

Gets the C1.Data.DataSource.ClientCollectionView of the RiaViewSource at the specified index in the ViewSources collection.

Syntax

Visual Basic (Declaration)

```
Public Overloads ReadOnly Property Item( _
   ByVal index As Integer _
) As ClientCollectionView
```

C#

```
public ClientCollectionView Item(
   int index
) {get;}
```

Parameters

index

The index of the RiaViewSource to take the C1.Data.DataSource.ClientCollectionView from.

Property Value

The C1.Data.DataSource.ClientCollectionView of the RiaViewSource.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members
Overload List

RefreshInterval Property
Gets or sets the interval between automatic Refresh operations to refresh the data with any changes that may have occurred on the server.

Syntax

Visual Basic (Declaration)
`Public Property RefreshInterval As TimeSpan`
C#
`public TimeSpan RefreshInterval {get; set;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

ViewSources Property
Gets a collection of RiaViewSource objects that define views (based on queries) in this C1DataSource.

Syntax

Visual Basic (Declaration)	
`Public ReadOnly Property ViewSources As RiaViewSourceCollection`	
C#	
`public RiaViewSourceCollection ViewSources {get;}`	

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

Fields
For a list of all members of this type, see C1DataSource members.

Public Fields

	Name	Description
S	ControlHandlerProperty	Identifies the C1.Silverlight.Data.RiaServices.C1DataSource.ControlHandler attached property.

Top

See Also

Reference

C1DataSource Class
C1.Silverlight.Data.RiaServices Namespace

ControlHandlerProperty Field
Identifies the C1.Silverlight.Data.RiaServices.C1DataSource.ControlHandler attached property.

Syntax

Visual Basic (Declaration)	
`Public Shared ReadOnly ControlHandlerProperty As DependencyProperty`	
C#	
`public static readonly DependencyProperty ControlHandlerProperty`	

Remarks

Use this attached property to connect a control handler to a control.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

Events
For a list of all members of this type, see C1DataSource members.

Public Events

	Name	Description
	BindingValidationError	(Inherited from System.Windows.FrameworkElement)
	DragEnter	(Inherited from System.Windows.UIElement)
	DragLeave	(Inherited from System.Windows.UIElement)
	DragOver	(Inherited from System.Windows.UIElement)

	Drop	(Inherited from System.Windows.UIElement)
	GotFocus	(Inherited from System.Windows.UIElement)
	IsEnabledChanged	(Inherited from System.Windows.Controls.Control)
	KeyDown	(Inherited from System.Windows.UIElement)
	KeyUp	(Inherited from System.Windows.UIElement)
	LayoutUpdated	(Inherited from System.Windows.FrameworkElement)
	Loaded	(Inherited from System.Windows.FrameworkElement)
	LostFocus	(Inherited from System.Windows.UIElement)
	LostMouseCapture	(Inherited from System.Windows.UIElement)
	MouseEnter	(Inherited from System.Windows.UIElement)
	MouseLeave	(Inherited from System.Windows.UIElement)
	MouseLeftButtonDown	(Inherited from System.Windows.UIElement)
	MouseLeftButtonUp	(Inherited from System.Windows.UIElement)
	MouseMove	(Inherited from System.Windows.UIElement)
	MouseRightButtonDown	(Inherited from System.Windows.UIElement)
	MouseRightButtonUp	(Inherited from System.Windows.UIElement)
	MouseWheel	(Inherited from System.Windows.UIElement)
	SavedChanges	Occurs after a submit operation is completed.

	SavingChanges	Occurs before changes are submitted.
	SizeChanged	(Inherited from System.Windows.FrameworkElement)
	TextInput	(Inherited from System.Windows.UIElement)
	TextInputStart	(Inherited from System.Windows.UIElement)
	TextInputUpdate	(Inherited from System.Windows.UIElement)
	Unloaded	(Inherited from System.Windows.FrameworkElement)

Top

See Also

Reference

C1DataSource Class
C1.Silverlight.Data.RiaServices Namespace

SavedChanges Event
Occurs after a submit operation is completed.

Syntax

Visual Basic (Declaration)
`Public Event SavedChanges As EventHandler(Of SavedChangesEventArgs)`
C#
`public event EventHandler<SavedChangesEventArgs> SavedChanges`

Event Data

The event handler receives an argument of type SavedChangesEventArgs containing data related to this event. The following **SavedChangesEventArgs** properties provide information specific to this event.

Property	Description

Error	Gets a value showing the error that occurred during a save operation.
HasError	Gets a value indicating whether the save operation has failed. If true, inspect the Error property for details.
IsErrorHandled	Gets a value indicating whether the error has been marked as handled by calling MarkErrorAsHandled.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members
SaveChanges Method

SavingChanges Event
Occurs before changes are submitted.

Syntax

Visual Basic (Declaration)
`Public Event SavingChanges As EventHandler(Of CancelEventArgs)`
C#
`public event EventHandler<CancelEventArgs> SavingChanges`

Event Data

The event handler receives an argument of type CancelEventArgs containing data related to this event. The following **CancelEventArgs** properties provide information specific to this event.

Property	Description

Cancel	

Remarks

This event is raised from the SaveChanges method and allows a handler to cancel the operation before it begins. When a handler sets System.ComponentModel.CancelEventArgs.Cancel to True, the operation will be aborted and a subsequent SavedChanges event will not be raised.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

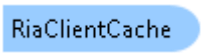
See Also

Reference

C1DataSource Class
C1DataSource Members
SaveChanges Method

RiaClientCache
Represents a client-side cache specific to RIA Services.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class RiaClientCache` `  Inherits C1.Data.ClientCacheBase`
C#
`public class RiaClientCache : C1.Data.ClientCacheBase`

Remarks

Usually, a single instance of this class is created on application startup and exists during the entire application lifetime, while each user control accesses the data using a RiaClientScope created by calling the CreateScope method.

Inheritance Hierarchy

System.Object
  C1.Data.ClientCacheBase
    C1.Silverlight.Data.RiaServices.RiaClientCache

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

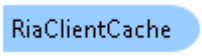
See Also

Reference

RiaClientCache Members
C1.Silverlight.Data.RiaServices Namespace

Overview
Represents a client-side cache specific to RIA Services.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class RiaClientCache` `  Inherits C1.Data.ClientCacheBase`
C#
`public class RiaClientCache : C1.Data.ClientCacheBase`

Remarks

Usually, a single instance of this class is created on application startup and exists during the entire application lifetime, while each user control accesses the data using a RiaClientScope created by calling the CreateScope method.

Inheritance Hierarchy

System.Object
  C1.Data.ClientCacheBase
    C1.Silverlight.Data.RiaServices.RiaClientCache

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

RiaClientCache Members
C1.Silverlight.Data.RiaServices Namespace

Members
Properties Methods

The following tables list the members exposed by RiaClientCache.

Public Constructors

	Name	Description
	RiaClientCache Constructor	Initializes a new instace of the RiaClientCache class.

Top

Public Properties

	Name	Description
	DomainContext	The System.ServiceModel.DomainServices.Client.DomainContext through which this RiaClientCache accesses the data.

	IsLoading	Gets a value indicating whether this RiaClientCache is currently fetching data from the server.

Top

Public Methods

	Name	Description
	BulkChanges	Used to group massive changes to entities and to allow manual explicit changes of entity states. (Inherited from C1.Data.ClientCacheBase)
	CleanupCache	Forces unused memory to be released, unused entities to be detached from the context. It is usually done automatically, so programmers rarely need to call this method in code. (Inherited from C1.Data.ClientCacheBase)
	Clear	Clears client-side cache entirely. Call this method if you want to make sure that following queries will fetch fresh data from the server. (Inherited from C1.Data.ClientCacheBase)
	CreateScope	Creates a RiaClientScope that defines a scope of data access.
	CreateTransaction	Creates a C1.Data.Transactions.ClientTransaction that allows you to easily cancel changes made in transaction scope. (Inherited from C1.Data.ClientCacheBase)
S	GetDefault	Returns the default RiaClientCache for a given contextType.
	Refresh	Refreshes data in all C1DataSource controls connected to this ClientCacheBase. (Inherited from C1.Data.ClientCacheBase)
S	RegisterContext	Overloaded. Registers a DomainContext as a default for C1DataSource controls for a given **context type**.

	RejectChanges	Reverts all pending changes for this C1.Data.ClientCacheBase. It is recommended to call this method instead of System.ServiceModel.DomainServices.Client.DomainContext.RejectChanges. (Inherited from C1.Data.ClientCacheBase)
	SaveChanges	Persists all changes to the server. This is a simple shortcut calling DomainContext.SubmitChanges. If you need more detailed control over the result of the submit changes operation, you can call the System.ServiceModel.DomainServices.Client.DomainContext.SubmitChanges method directly. (Inherited from C1.Data.ClientCacheBase)

Top

See Also

Reference

RiaClientCache Class
C1.Silverlight.Data.RiaServices Namespace

RiaClientCache Constructor
The DomainContext used to access the data.

Initializes a new instace of the RiaClientCache class.

Syntax

Visual Basic (Declaration)

```
Public Function New( _
   ByVal baseContext As DomainContext _
)
```

C#

```
public RiaClientCache(
   DomainContext baseContext
)
```

Parameters

baseContext

The DomainContext used to access the data.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

RiaClientCache Class
RiaClientCache Members

Methods
For a list of all members of this type, see RiaClientCache members.

Public Methods

	Name	Description
	BulkChanges	Used to group massive changes to entities and to allow manual explicit changes of entity states. (Inherited from C1.Data.ClientCacheBase)
	CleanupCache	Forces unused memory to be released, unused entities to be detached from the context. It is usually done automatically, so programmers rarely need to call this method in code. (Inherited from C1.Data.ClientCacheBase)
	Clear	Clears client-side cache entirely. Call this method if you want to make sure that following queries will fetch fresh data from the server. (Inherited from C1.Data.ClientCacheBase)
	CreateScope	Creates a RiaClientScope that defines a scope of data access.
	CreateTransaction	Creates a C1.Data.Transactions.ClientTransaction that allows you to easily cancel changes made in transaction scope. (Inherited from C1.Data.ClientCacheBase)

S	GetDefault	Returns the default RiaClientCache for a given contextType.
	Refresh	Refreshes data in all C1DataSource controls connected to this ClientCacheBase. (Inherited from C1.Data.ClientCacheBase)
S	RegisterContext	Overloaded. Registers a DomainContext as a default for C1DataSource controls for a given **context type**.
	RejectChanges	Reverts all pending changes for this C1.Data.ClientCacheBase. It is recommended to call this method instead of System.ServiceModel.DomainServices.Client.DomainContext.RejectChanges. (Inherited from C1.Data.ClientCacheBase)
	SaveChanges	Persists all changes to the server. This is a simple shortcut calling DomainContext.SubmitChanges. If you need more detailed control over the result of the submit changes operation, you can call the System.ServiceModel.DomainServices.Client.DomainContext.SubmitChanges method directly. (Inherited from C1.Data.ClientCacheBase)

Top

See Also

Reference

RiaClientCache Class
C1.Silverlight.Data.RiaServices Namespace

CreateScope Method
Creates a RiaClientScope that defines a scope of data access.

Syntax

Visual Basic (Declaration)
`Public Shadows Function CreateScope() As RiaClientScope`
C#

```
public new RiaClientScope CreateScope()
```

Return Value

A new client scope.

Remarks

Usually, each user control creates a RiaClientScope and uses it to access entities.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

RiaClientCache Class
RiaClientCache Members
RiaClientScope Class

GetDefault Method
A subclass of DomainContext to get the default RiaClientCache for.

Returns the default RiaClientCache for a given contextType.

Syntax

Visual Basic (Declaration)

```
Public Shared Function GetDefault( _
   ByVal contextType As Type _
) As RiaClientCache
```

C#

```
public static RiaClientCache GetDefault(
   Type contextType
)
```

Parameters

contextType

A subclass of DomainContext to get the default RiaClientCache for.

Return Value

The default RiaClientCache for the specified contextType.

Remarks

Creates an RiaClientCache for the specified contextType if it does not already exist; otherwise, returns an existing instance.

It is the same default client cache as used by C1DataSource with specified C1DataSource.DomainContextTypeName.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

RiaClientCache Class
RiaClientCache Members

RegisterContext Method
Registers a DomainContext as a default for C1DataSource controls for a given **context type**.

Overload List

Overload	Description
RegisterContext(DomainContext,Type)	Registers a DomainContext as a default for C1DataSource controls for a given contextType.
RegisterContext(DomainContext)	Registers a DomainContext as a default for C1DataSource controls.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

RiaClientCache Class
RiaClientCache Members

RegisterContext(DomainContext,Type) Method
A DomainContext to set as default.

The type (derived from DomainContext) to register the context for.

Registers a DomainContext as a default for C1DataSource controls for a given contextType.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function RegisterContext( _
   ByVal context As DomainContext, _
   ByVal contextType As Type _
) As IDisposable
```

C#

```
public static IDisposable RegisterContext(
   DomainContext context,
   Type contextType
)
```

Parameters

context

A DomainContext to set as default.

contextType

The type (derived from DomainContext) to register the context for.

Return Value

An System.IDisposable to unregister the context.

Exceptions

Exception	Description
System.InvalidOperationException	Another context is already registered for the given contextType.

Remarks

Use this method when you need to customize the default DomainContext used in C1DataSource controls. Register a custom DomainContext on startup before any C1DataSource instances are created.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

RiaClientCache Class
RiaClientCache Members
Overload List
GetDefault Method

RegisterContext(DomainContext) Method
A DomainContext to set as default.

Registers a DomainContext as a default for C1DataSource controls.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function RegisterContext( _
   ByVal context As DomainContext _
) As IDisposable
```

C#

```
public static IDisposable RegisterContext(
   DomainContext context
)
```

Parameters

context

A DomainContext to set as default.

Return Value

An System.IDisposable to unregister the context.

Exceptions

Exception	Description
System.InvalidOperationException	Another context is already registered.

Remarks

Use this method when you need to customize the default DomainContext used in C1DataSource controls. Register a custom DomainContext on startup before any C1DataSource instances are created.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

RiaClientCache Class
RiaClientCache Members
Overload List
GetDefault Method

Properties
For a list of all members of this type, see RiaClientCache members.

Public Properties

	Name	Description
	DomainContext	The System.ServiceModel.DomainServices.Client.DomainContext through which this RiaClientCache accesses the data.
	IsLoading	Gets a value indicating whether this RiaClientCache is currently fetching data from the server.

Top

See Also

Reference

RiaClientCache Class
C1.Silverlight.Data.RiaServices Namespace

DomainContext Property
The System.ServiceModel.DomainServices.Client.DomainContext through which this RiaClientCache accesses the data.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Property DomainContext As DomainContext
```

C#

```
public DomainContext DomainContext {get;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

RiaClientCache Class
RiaClientCache Members

IsLoading Property
Gets a value indicating whether this RiaClientCache is currently fetching data from the server.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property IsLoading As Boolean`
C#
`public bool IsLoading {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

RiaClientCache Class
RiaClientCache Members

RiaClientScope
Defines a scope of data access. Provides facilities to create client views.

Object Model

Syntax

Visual Basic (Declaration)
`Public Class RiaClientScope`

```
Inherits C1.Data.ClientScope
```

C#

```
public class RiaClientScope : C1.Data.ClientScope
```

Remarks

Usually, one scope is created for each user control, and disposed with the user control. Entities pinned to the scope (marked as needed) are not evicted from the cache until the scope is disposed or collected by the garbage collector.

Inheritance Hierarchy

System.Object
  C1.Data.ClientScope
    C1.Silverlight.Data.RiaServices.RiaClientScope

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

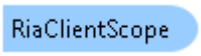
See Also

Reference

RiaClientScope Members
C1.Silverlight.Data.RiaServices Namespace

Overview
Defines a scope of data access. Provides facilities to create client views.

Object Model



Syntax

Visual Basic (Declaration)

```
Public Class RiaClientScope
```

```
Inherits C1.Data.ClientScope
```

C#

```
public class RiaClientScope : C1.Data.ClientScope
```

Remarks

Usually, one scope is created for each user control, and disposed with the user control. Entities pinned to the scope (marked as needed) are not evicted from the cache until the scope is disposed or collected by the garbage collector.

Inheritance Hierarchy

System.Object
  C1.Data.ClientScope
    C1.Silverlight.Data.RiaServices.RiaClientScope

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

RiaClientScope Members
C1.Silverlight.Data.RiaServices Namespace

Members
Properties Methods

The following tables list the members exposed by RiaClientScope.

Public Constructors

	Name	Description
	RiaClientScope Constructor	Initializes a new instance of the RiaClientScope class with the specified RiaClientCache.

Top

Public Properties

	Name	Description
	ClientCache	Gets the RiaClientCache this client scope is connected to.

Top

Public Methods

	Name	Description
	AddRef	Overloaded. Marks an entity as needed. Needed entities are not detached/released from the context until the client scope is disposed. (Inherited from C1.Data.ClientScope)
	Dispose	Marks the scope as disposed. Entities that were marked needed by a disposed scope may be disposed of (evicted from the cache, detached from context) unless they are needed by other active scopes. (Inherited from C1.Data.ClientScope)
	GetItems	Overloaded. Gets a client view of entities loaded using a given **entity query**.
	Release	Overloaded. Unmark a needed entity. (Inherited from C1.Data.ClientScope)

Top

See Also

Reference

RiaClientScope Class
C1.Silverlight.Data.RiaServices Namespace

RiaClientScope Constructor
An instance of the RiaClientCache class to which the new client scope is connected.

Initializes a new instance of the RiaClientScope class with the specified RiaClientCache.

Syntax

Visual Basic (Declaration)

```
Public Function New( _
   ByVal clientCache As RiaClientCache _
)
```

C#

```
public RiaClientScope(
   RiaClientCache clientCache
)
```

Parameters

clientCache

An instance of the RiaClientCache class to which the new client scope is connected.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

RiaClientScope Class
RiaClientScope Members

Methods
For a list of all members of this type, see RiaClientScope members.

Public Methods

	Name	Description
	AddRef	Overloaded. Marks an entity as needed. Needed entities are not detached/released from the context until the client scope is disposed.

		(Inherited from C1.Data.ClientScope)
	Dispose	Marks the scope as disposed. Entities that were marked needed by a disposed scope may be disposed of (evicted from the cache, detached from context) unless they are needed by other active scopes. (Inherited from C1.Data.ClientScope)
	GetItems	Overloaded. Gets a client view of entities loaded using a given **entity query**.
	Release	Overloaded. Unmark a needed entity. (Inherited from C1.Data.ClientScope)

Top

See Also

Reference

RiaClientScope Class
C1.Silverlight.Data.RiaServices Namespace

GetItems Method
Gets a client view of entities loaded using a given **entity query**.

Overload List

Overload	Description
GetItems<T>(String,IDictionary<String,Object>)	Gets a client view of entities loaded using a given queryName.
GetItems<T>(EntityQuery<T>)	Gets a client view of entities loaded using a given query.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

RiaClientScope Class
RiaClientScope Members

GetItems<T>(String,IDictionary<String,Object>) Method
The type of entities returned by the queryName.

The name of the query used to load entities.

Optional query parameters for the queryName.

Gets a client view of entities loaded using a given queryName.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function GetItems(Of T As Entity)( _
   ByVal queryName As String, _
   Optional ByVal parameters As IDictionary(Of String,Object) _
) As ClientView(Of T)
```

C#

```
public ClientView<T> GetItems<T>(
   string queryName,
   IDictionary<string,object> parameters
)
where T: Entity
```

Parameters

queryName

The name of the query used to load entities.

parameters

Optional query parameters for the queryName.

Type Parameters

T

The type of entities returned by the queryName.

Return Value

A client view of entities loaded by the specified queryName.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

RiaClientScope Class
RiaClientScope Members
Overload List

GetItems<T>(EntityQuery<T>) Method
The type of entities returned by the query.

The query used to load entities.

Gets a client view of entities loaded using a given query.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function GetItems(Of T As Entity)( _
   ByVal query As EntityQuery(Of T) _
) As ClientView(Of T)
```

C#

```
public ClientView<T> GetItems<T>(
   EntityQuery<T> query
)
where T: Entity
```

Parameters

query

The query used to load entities.

Type Parameters

T

The type of entities returned by the query.

Return Value

A client view of entities loaded by the specified query.

Remarks

The query must not be changed after using this method.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

RiaClientScope Class
RiaClientScope Members
Overload List

Properties
For a list of all members of this type, see RiaClientScope members.

Public Properties

	Name	Description
	ClientCache	Gets the RiaClientCache this client scope is connected to.

Top

See Also

Reference

RiaClientScope Class
C1.Silverlight.Data.RiaServices Namespace

ClientCache Property
Gets the RiaClientCache this client scope is connected to.

Syntax

Visual Basic (Declaration)
`Public Shadows ReadOnly Property ClientCache As RiaClientCache`
C#
`public new RiaClientCache ClientCache {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

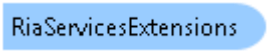
See Also

Reference

RiaClientScope Class
RiaClientScope Members

RiaServicesExtensions
Provides a set of extensions methods for RIA Services.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit NotInheritable Class RiaServicesExtensions`
C#

```
public static class RiaServicesExtensions
```

Inheritance Hierarchy

System.Object
  C1.Silverlight.Data.RiaServices.RiaServicesExtensions

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

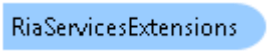
See Also

Reference

RiaServicesExtensions Members
C1.Silverlight.Data.RiaServices Namespace

Overview
Provides a set of extensions methods for RIA Services.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit NotInheritable Class RiaServicesExtensions`
C#
`public static class RiaServicesExtensions`

Inheritance Hierarchy

System.Object
  C1.Silverlight.Data.RiaServices.RiaServicesExtensions

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

RiaServicesExtensions Members
C1.Silverlight.Data.RiaServices Namespace

Members
Methods

The following tables list the members exposed by RiaServicesExtensions.

Public Methods

	Name	Description
S	AsLive<T>	Converts a given entities to a live view. That live view also supports adding/removing entities to/from the given clientCache.

Top

See Also

Reference

RiaServicesExtensions Class
C1.Silverlight.Data.RiaServices Namespace

Methods
For a list of all members of this type, see RiaServicesExtensions members.

Public Methods

	Name	Description
S	AsLive<T>	Converts a given entities to a live view. That live view also supports adding/removing entities to/from the given clientCache.

Top

See Also

Reference

RiaServicesExtensions Class
C1.Silverlight.Data.RiaServices Namespace

AsLive<T> Method
The type of the entities in the given entities.

The entity collection to convert.

An instance of the RiaClientCache class to which this entity collection belongs. It is used in adding/removing entities.

Converts a given entities to a live view. That live view also supports adding/removing entities to/from the given clientCache.

Syntax

Visual Basic (Declaration)

```
Public Shared Function AsLive(Of T As Entity)( _
   ByVal entities As EntityCollection(Of T), _
   ByVal clientCache As RiaClientCache _
) As View(Of T)
```

C#

```
public static View<T> AsLive<T>(
   EntityCollection<T> entities,
   RiaClientCache clientCache
)
where T: Entity
```

Parameters

entities

The entity collection to convert.

clientCache

An instance of the RiaClientCache class to which this entity collection belongs. It is used in adding/removing entities.

Type Parameters

T

The type of the entities in the given entities.

Return Value

The resulting live view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

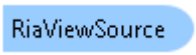
See Also

Reference

RiaServicesExtensions Class
RiaServicesExtensions Members

RiaViewSource
A RIA Services-specific version of the C1.Data.DataSource.ClientViewSource class.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class RiaViewSource` `  Inherits C1.Data.DataSource.ClientViewSource`
C#
`public class RiaViewSource : C1.Data.DataSource.ClientViewSource`

Remarks

To load data, set the QueryName to the name of the method in the System.ServiceModel.DomainServices.Client.DomainContext that returns an

System.ServiceModel.DomainServices.Client.EntityQuery{T}. Use Parameters to specify query parameters.

Inheritance Hierarchy

System.Object
  System.Windows.DependencyObject
    C1.Data.DataSource.ClientViewSource
      C1.Silverlight.Data.RiaServices.RiaViewSource

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

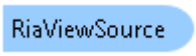
See Also

Reference

RiaViewSource Members
C1.Silverlight.Data.RiaServices Namespace

Overview
A RIA Services-specific version of the C1.Data.DataSource.ClientViewSource class.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class RiaViewSource` `  Inherits C1.Data.DataSource.ClientViewSource`
C#
`public class RiaViewSource : C1.Data.DataSource.ClientViewSource`

Remarks

To load data, set the QueryName to the name of the method in the System.ServiceModel.DomainServices.Client.DomainContext that returns an System.ServiceModel.DomainServices.Client.EntityQuery{T}. Use Parameters to specify query parameters.

Inheritance Hierarchy

System.Object
  System.Windows.DependencyObject
    C1.Data.DataSource.ClientViewSource
      C1.Silverlight.Data.RiaServices.RiaViewSource

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

RiaViewSource Members
C1.Silverlight.Data.RiaServices Namespace

Members
Properties Methods Events

The following tables list the members exposed by RiaViewSource.

Public Constructors

	Name	Description
	RiaViewSource Constructor	Initializes a new instance of the RiaViewSource class.

Top

Public Properties

	Name	Description

	AutoLoad	Gets or sets a value indicating whether Load is automatically invoked on startup and when a change occurs that impacts the query composed by the C1.Data.DataSource.ClientViewSource. The default is True. (Inherited from C1.Data.DataSource.ClientViewSource)
	BaseView	Gets or sets an instance of C1.Data.ClientView<T> that the C1.Data.DataSource.ClientViewSource uses as the base for composing queries. (Inherited from C1.Data.DataSource.ClientViewSource)
	CacheTimeout	Gets or sets the period of time entities loaded in virtual mode are kept in the cache without checking whether they are needed or not. If an entity was neither used nor considered needed for a period of time longer than CacheTimeout, C1.Data.DataSource.ClientViewSource may evict it from the cache. (Inherited from C1.Data.DataSource.ClientViewSource)
	CurrentClientView	Gets the current client view used to load entities, or null in virtual mode. (Inherited from C1.Data.DataSource.ClientViewSource)
	DataView	Gets the current view of entities resulting from the last load operation. (Inherited from C1.Data.DataSource.ClientViewSource)
	Dispatcher	(Inherited from System.Windows.DependencyObject)
	FilterDescriptors	Gets the collection of System.Windows.Controls.FilterDescriptor objects used when performing loads. (Inherited from C1.Data.DataSource.ClientViewSource)
	FilterOperator	Gets or sets the logical operator used for combinining FilterDescriptors in the filter collection. The default value is System.Windows.Controls.FilterDescriptorLogicalOperator.And. (Inherited from C1.Data.DataSource.ClientViewSource)
	GroupDescriptors	Gets the collection of System.Windows.Controls.GroupDescriptor objects used to organize the loaded entities into groups. (Inherited from

		C1.Data.DataSource.ClientViewSource)
	IsLoadingData	Gets a value indicating whether the C1.Data.DataSource.ClientViewSource is currently loading data. (Inherited from C1.Data.DataSource.ClientViewSource)
	LoadCommand	Gets an System.Windows.Input.ICommand that invokes Load on this C1.Data.DataSource.ClientViewSource. (Inherited from C1.Data.DataSource.ClientViewSource)
	LoadDelay	Gets or sets the delay before an automatic data loading operation is started. It is the delay from the time a change prompting automatic load occurs until the time the resulting Load is started. The default delay is 25 milliseconds. (Inherited from C1.Data.DataSource.ClientViewSource)
	LoadSize	Gets or sets the maximum number of items to load each time a Load is executed. When equal to 0, all requested entities will be loaded. The default is 0. (Inherited from C1.Data.DataSource.ClientViewSource)
	MoveToFirstOnLoad	Gets or sets a value indicating that the first item must be made current after Load operation is completed if current item was not set by other means. (Inherited from C1.Data.DataSource.ClientViewSource)
	Name	Overridden. Gets a name of this RiaViewSource to reference it in a C1DataSource.ViewSources collection. It is determined by the QueryName but can be overrided by the NameOverride.
	NameOverride	Gets or sets a value that overrides the value of the Name property. (Inherited from C1.Data.DataSource.ClientViewSource)
	PageSize	Gets or sets the number of items displayed on each page of the DataView, or the number of items to fetch in each query in virtual mode, or 0 to disable paging. (Inherited from C1.Data.DataSource.ClientViewSource)

	Parameters	Gets a collection of parameters for the System.ServiceModel.DomainServices.Client.EntityQuery{T} specified by the QueryName.
	QueryName	Gets or sets the name of an System.ServiceModel.DomainServices.Client.EntityQuery{T} to be used a source.
	SortDescriptors	Gets the collection of System.Windows.Controls.SortDescriptor objects used to sort the data. (Inherited from C1.Data.DataSource.ClientViewSource)
	VirtualMode	Gets or sets a value indicating whether the C1.Data.DataSource.ClientViewSource is in virtual mode. Virtual mode is an innovative technology allowing to bind GUI controls directly to very large data sets without delays and performance degradation and without inconvenience of paging. By default, virtual mode is disabled (the default value is C1.Data.DataSource.VirtualModeKind.None). (Inherited from C1.Data.DataSource.ClientViewSource)

Top

Public Methods

	Name	Description
	ClearValue	(Inherited from System.Windows.DependencyObject)
	DeferLoad	Used to group changes to multiple load-affecting properties together, deferring the resulting load operations so a single load operation is performed in the end, that is, when the object returned from this method is disposed. (Inherited from C1.Data.DataSource.ClientViewSource)
	GetAnimationBaseValue	(Inherited from System.Windows.DependencyObject)

	GetValue	(Inherited from System.Windows.DependencyObject)
	Load	Starts a load operation. Any pending load will be implicitly canceled. (Inherited from C1.Data.DataSource.ClientViewSource)
	LoadRange	If in virtual mode, loads a specific range of entities. (Inherited from C1.Data.DataSource.ClientViewSource)
	ReadLocalValue	(Inherited from System.Windows.DependencyObject)
	Refresh	Starts a load operation ignoring the client-side cache. Any pending load will be implicitly canceled. (Inherited from C1.Data.DataSource.ClientViewSource)
	SetValue	(Inherited from System.Windows.DependencyObject)

Top

Public Events

	Name	Description
	LoadedData	Occurs when a load operation is completed, or when an exception was thrown during the load operation. (Inherited from C1.Data.DataSource.ClientViewSource)
	PropertyChanged	Occurs when a property value changes. (Inherited from C1.Data.DataSource.ClientViewSource)

Top

See Also

Reference

RiaViewSource Class
C1.Silverlight.Data.RiaServices Namespace

RiaViewSource Constructor
Initializes a new instance of the RiaViewSource class.

Syntax

Visual Basic (Declaration)
`Public Function New()`
C#
`public RiaViewSource()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

RiaViewSource Class
RiaViewSource Members

Properties
For a list of all members of this type, see RiaViewSource members.

Public Properties

	Name	Description
	AutoLoad	Gets or sets a value indicating whether Load is automatically invoked on startup and when a change occurs that impacts the query composed by the C1.Data.DataSource.ClientViewSource. The default is True. (Inherited from C1.Data.DataSource.ClientViewSource)
	BaseView	Gets or sets an instance of C1.Data.ClientView<T> that the C1.Data.DataSource.ClientViewSource uses as the base for composing queries. (Inherited from C1.Data.DataSource.ClientViewSource)

	CacheTimeout	Gets or sets the period of time entities loaded in virtual mode are kept in the cache without checking whether they are needed or not. If an entity was neither used nor considered needed for a period of time longer than CacheTimeout, C1.Data.DataSource.ClientViewSource may evict it from the cache. (Inherited from C1.Data.DataSource.ClientViewSource)
	CurrentClientView	Gets the current client view used to load entities, or null in virtual mode. (Inherited from C1.Data.DataSource.ClientViewSource)
	DataView	Gets the current view of entities resulting from the last load operation. (Inherited from C1.Data.DataSource.ClientViewSource)
	Dispatcher	(Inherited from System.Windows.DependencyObject)
	FilterDescriptors	Gets the collection of System.Windows.Controls.FilterDescriptor objects used when performing loads. (Inherited from C1.Data.DataSource.ClientViewSource)
	FilterOperator	Gets or sets the logical operator used for combinining FilterDescriptors in the filter collection. The default value is System.Windows.Controls.FilterDescriptorLogicalOperator.And. (Inherited from C1.Data.DataSource.ClientViewSource)
	GroupDescriptors	Gets the collection of System.Windows.Controls.GroupDescriptor objects used to organize the loaded entities into groups. (Inherited from C1.Data.DataSource.ClientViewSource)
	IsLoadingData	Gets a value indicating whether the C1.Data.DataSource.ClientViewSource is currently loading data. (Inherited from C1.Data.DataSource.ClientViewSource)
	LoadCommand	Gets an System.Windows.Input.ICommand that invokes Load on this C1.Data.DataSource.ClientViewSource. (Inherited from C1.Data.DataSource.ClientViewSource)

	LoadDelay	Gets or sets the delay before an automatic data loading operation is started. It is the delay from the time a change prompting automatic load occurs until the time the resulting Load is started. The default delay is 25 milliseconds. (Inherited from C1.Data.DataSource.ClientViewSource)
	LoadSize	Gets or sets the maximum number of items to load each time a Load is executed. When equal to 0, all requested entities will be loaded. The default is 0. (Inherited from C1.Data.DataSource.ClientViewSource)
	MoveToFirstOnLoad	Gets or sets a value indicating that the first item must be made current after Load operation is completed if current item was not set by other means. (Inherited from C1.Data.DataSource.ClientViewSource)
	Name	Overridden. Gets a name of this RiaViewSource to reference it in a C1DataSource.ViewSources collection. It is determined by the QueryName but can be overrided by the NameOverride.
	NameOverride	Gets or sets a value that overrides the value of the Name property. (Inherited from C1.Data.DataSource.ClientViewSource)
	PageSize	Gets or sets the number of items displayed on each page of the DataView, or the number of items to fetch in each query in virtual mode, or 0 to disable paging. (Inherited from C1.Data.DataSource.ClientViewSource)
	Parameters	Gets a collection of parameters for the System.ServiceModel.DomainServices.Client.EntityQuery{T} specified by the QueryName.
	QueryName	Gets or sets the name of an System.ServiceModel.DomainServices.Client.EntityQuery{T} to be used a source.
	SortDescriptors	Gets the collection of System.Windows.Controls.SortDescriptor objects used to sort the data. (Inherited from

		C1.Data.DataSource.ClientViewSource)
	VirtualMode	Gets or sets a value indicating whether the C1.Data.DataSource.ClientViewSource is in virtual mode. Virtual mode is an innovative technology allowing to bind GUI controls directly to very large data sets without delays and performance degradation and without inconvenience of paging. By default, virtual mode is disabled (the default value is C1.Data.DataSource.VirtualModeKind.None). (Inherited from C1.Data.DataSource.ClientViewSource)

Top

See Also

Reference

RiaViewSource Class
C1.Silverlight.Data.RiaServices Namespace

Name Property
Gets a name of this RiaViewSource to reference it in a C1DataSource.ViewSources collection. It is determined by the QueryName but can be overrided by the NameOverride.

Syntax

Visual Basic (Declaration)

```
Public Overrides ReadOnly Property Name As String
```

C#

```
public override string Name {get;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

RiaViewSource Class
RiaViewSource Members

Parameters Property
Gets a collection of parameters for the System.ServiceModel.DomainServices.Client.EntityQuery{T} specified by the QueryName.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Parameters As ParameterCollection`
C#
`public ParameterCollection Parameters {get;}`

Remarks

Changing this collection or properties of System.Windows.Controls.Parameter objects in this collection causes the RiaViewSource to reload data if AutoLoad is set to true.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

RiaViewSource Class
RiaViewSource Members

QueryName Property
Gets or sets the name of an System.ServiceModel.DomainServices.Client.EntityQuery{T} to be used a source.

Syntax

Visual Basic (Declaration)
`Public Property QueryName As String`

C#

```
public string QueryName {get; set;}
```

Remarks

Changing the value of this property causes the RiaViewSource to reload data if AutoLoad is set to true.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

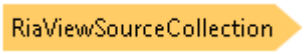
See Also

Reference

RiaViewSource Class
RiaViewSource Members

RiaViewSourceCollection
An observable collection of RiaViewSource objects.

Object Model



Syntax

Visual Basic (Declaration)

```
Public Class RiaViewSourceCollection
   Inherits System.Collections.ObjectModel.ObservableCollection(Of
RiaViewSource)
```

C#

```
public class RiaViewSourceCollection :
System.Collections.ObjectModel.ObservableCollection<RiaViewSource>
```

Inheritance Hierarchy

System.Object
  System.Collections.ObjectModel.Collection<T>
    System.Collections.ObjectModel.ObservableCollection<T>
      C1.Silverlight.Data.RiaServices.RiaViewSourceCollection

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

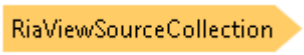
See Also

Reference

RiaViewSourceCollection Members
C1.Silverlight.Data.RiaServices Namespace

Overview
An observable collection of RiaViewSource objects.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class RiaViewSourceCollection` `  Inherits System.Collections.ObjectModel.ObservableCollection(Of RiaViewSource)`
C#
`public class RiaViewSourceCollection : System.Collections.ObjectModel.ObservableCollection<RiaViewSource>`

Inheritance Hierarchy

System.Object
  System.Collections.ObjectModel.Collection<T>

System.Collections.ObjectModel.ObservableCollection<T>
C1.Silverlight.Data.RiaServices.RiaViewSourceCollection

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

RiaViewSourceCollection Members
C1.Silverlight.Data.RiaServices Namespace

Members
Properties Methods Events

The following tables list the members exposed by RiaViewSourceCollection.

Public Constructors

	Name	Description
	RiaViewSourceCollection Constructor	

Top

Public Properties

	Name	Description
	Count	(Inherited from System.Collections.ObjectModel.Collection<RiaViewSource>)
	Item	Gets the RiaViewSource with the specified name.

Top

Public Methods

	Name	Description
	Add	(Inherited from System.Collections.ObjectModel.Collection<RiaViewSource>)
	Clear	(Inherited from System.Collections.ObjectModel.Collection<RiaViewSource>)
	Contains	(Inherited from System.Collections.ObjectModel.Collection<RiaViewSource>)
	CopyTo	(Inherited from System.Collections.ObjectModel.Collection<RiaViewSource>)
	GetEnumerator	(Inherited from System.Collections.ObjectModel.Collection<RiaViewSource>)
	IndexOf	(Inherited from System.Collections.ObjectModel.Collection<RiaViewSource>)
	Insert	(Inherited from System.Collections.ObjectModel.Collection<RiaViewSource>)
	Remove	(Inherited from System.Collections.ObjectModel.Collection<RiaViewSource>)
	RemoveAt	(Inherited from System.Collections.ObjectModel.Collection<RiaViewSource>)

Top

Public Events

	Name	Description
	CollectionChanged	(Inherited from System.Collections.ObjectModel.ObservableCollection<RiaViewSource>)

Top

See Also

Reference

RiaViewSourceCollection Class
C1.Silverlight.Data.RiaServices Namespace

RiaViewSourceCollection Constructor

Syntax

Visual Basic (Declaration)
`Public Function New()`
C#
`public RiaViewSourceCollection()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

RiaViewSourceCollection Class
RiaViewSourceCollection Members

Properties
For a list of all members of this type, see RiaViewSourceCollection members.

Public Properties

	Name	Description
	Count	(Inherited from System.Collections.ObjectModel.Collection<RiaViewSource>)
	Item	Gets the RiaViewSource with the specified name.

Top

See Also

Reference

RiaViewSourceCollection Class
C1.Silverlight.Data.RiaServices Namespace

Item Property

The name of the RiaViewSource to get from the collection.

Gets the RiaViewSource with the specified name.

Syntax

Visual Basic (Declaration)

`Public Shadows ReadOnly Default Property Item( _`
`  ByVal name As String _`
`) As RiaViewSource`

C#

`public new RiaViewSource this[`
`  string name`
`]; {get;}`

Parameters

name

The name of the RiaViewSource to get from the collection.

Property Value

The RiaViewSource with the specified name, or null if it does not exist.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

RiaViewSourceCollection Class
RiaViewSourceCollection Members

C1.Util.Licensing Namespace

Overview

Classes

	Class	Description
	LicenseMode	Provides the licensing mode connection between a control and its designer.

See Also

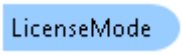
Reference

C1.Silverlight.Data.Entity Assembly

Classes

LicenseMode
Provides the licensing mode connection between a control and its designer.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class LicenseMode`
C#
`public class LicenseMode`

Inheritance Hierarchy

System.Object
  C1.Util.Licensing.LicenseMode

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

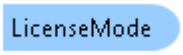
See Also

Reference

LicenseMode Members
C1.Util.Licensing Namespace

Overview
Provides the licensing mode connection between a control and its designer.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class LicenseMode`
C#
`public class LicenseMode`

Inheritance Hierarchy

System.Object
  C1.Util.Licensing.LicenseMode

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LicenseMode Members
C1.Util.Licensing Namespace

Members
Fields Methods

The following tables list the members exposed by LicenseMode.

Public Constructors

	Name	Description
	LicenseMode Constructor	

Top

Public Fields

	Name	Description
S	EvaluationProperty	This property is for internal use only.

Top

Public Methods

	Name	Description
S	GetEvaluation	This property is for internal use only.
S	SetEvaluation	This property is for internal use only.

Top

See Also

Reference

LicenseMode Class
C1.Util.Licensing Namespace

LicenseMode Constructor

Syntax

Visual Basic (Declaration)
`Public Function New()`
C#
`public LicenseMode()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LicenseMode Class
LicenseMode Members

Methods
>

Name	Description
GetEvaluation	This property is for internal use only.
SetEvaluation	This property is for internal use only.

Top

See Also

Reference

LicenseMode Class
C1.Util.Licensing Namespace

GetEvaluation Method
This property is for internal use only.

Syntax

Visual Basic (Declaration)

```
Public Shared Function GetEvaluation( _
   ByVal obj As DependencyObject _
) As Boolean
```

C#

```
public static bool GetEvaluation(
   DependencyObject obj
)
```

Parameters

obj

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LicenseMode Class
LicenseMode Members

SetEvaluation Method
This property is for internal use only.

Syntax

Visual Basic (Declaration)

```
Public Shared Sub SetEvaluation( _
   ByVal obj As DependencyObject, _
   ByVal value As Boolean _
)
```

C#

```
public static void SetEvaluation(
   DependencyObject obj,
   bool value
)
```

Parameters

obj

value

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LicenseMode Class
LicenseMode Members

Fields
For a list of all members of this type, see LicenseMode members.

Public Fields

	Name	Description
S	EvaluationProperty	This property is for internal use only.

Top

See Also

Reference

LicenseMode Class
C1.Util.Licensing Namespace

EvaluationProperty Field

This property is for internal use only.

Syntax

Visual Basic (Declaration)
`Public Shared ReadOnly EvaluationProperty As DependencyProperty`
C#
`public static readonly DependencyProperty EvaluationProperty`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LicenseMode Class
LicenseMode Members

C1.Silverlight.LiveLinq Assembly

Overview

%%description%%

" -->

Namespaces

Namespace	Description
(Global)	
C1.LiveLinq	
C1.LiveLinq.LiveViews	

C1.LiveLinq.LiveViews.Xml	
C1.LiveLinq.Metadata	

Namespaces

(Global) Namespace

Overview

Classes

	Class	Description
	C1LiveLinqInfo	Provides information related to the C1.LiveLinq.dll assembly.

See Also

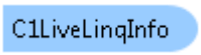
Reference

C1.Silverlight.LiveLinq Assembly

Classes

C1LiveLinqInfo
Provides information related to the C1.LiveLinq.dll assembly.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit NotInheritable Class C1LiveLinqInfo`
C#
`public static class C1LiveLinqInfo`

Inheritance Hierarchy

System.Object
  C1LiveLinqInfo

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

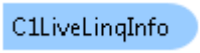
See Also

Reference

C1LiveLinqInfo Members
(Global) Namespace

Overview
Provides information related to the C1.LiveLinq.dll assembly.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit NotInheritable Class C1LiveLinqInfo`
C#
`public static class C1LiveLinqInfo`

Inheritance Hierarchy

System.Object
  C1LiveLinqInfo

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1LiveLinqInfo Members
(Global) Namespace

Members
Fields

The following tables list the members exposed by C1LiveLinqInfo.

Public Fields

	Name	Description
	PublicKey_C1_Silverlight_LiveLinq	Contains the public key of the C1.Silverlight.LiveLinq.dll.
	PublicKey_System_Core	Contains the public key of the System.Core.dll.

Top

See Also

Reference

C1LiveLinqInfo Class
(Global) Namespace

Fields
For a list of all members of this type, see C1LiveLinqInfo members.

Public Fields

	Name	Description
	PublicKey_C1_Silverlight_LiveLinq	Contains the public key of the C1.Silverlight.LiveLinq.dll.
	PublicKey_System_Core	Contains the public key of the System.Core.dll.

Top

See Also

Reference

C1LiveLinqInfo Class
(Global) Namespace

PublicKey_C1_Silverlight_LiveLinq Field
Example

Contains the public key of the C1.Silverlight.LiveLinq.dll.

Syntax

Visual Basic (Declaration)	
`Public Const PublicKey_C1_Silverlight_LiveLinq As String`	
C#	
`public const string PublicKey_C1_Silverlight_LiveLinq`	

Example

- C#

```
[assembly:
System.Runtime.CompilerServices.InternalsVisibleTo("C1.Silverlight.LiveLinq,
PublicKey=" + C1LiveLinqInfo.PublicKey_C1_Silverlight_LiveLinq)]
```

Remarks

Use this constant to make your internal members visible to C1.Silverlight.LiveLinq.dll in order to prevent System.MemberAccessException exceptions.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

PublicKey_System_Core Field
Example

Contains the public key of the System.Core.dll.

Syntax

Visual Basic (Declaration)
`Public Const PublicKey_System_Core As String`
C#
`public const string PublicKey_System_Core`

Example

- C#

```
[assembly:
System.Runtime.CompilerServices.InternalsVisibleToAttribute("System.Core,
PublicKey=" + C1LiveLinqInfo.PublicKey_System_Core)]
```

Remarks

Use this constant to make your internal members visible to System.Core.dll in order to prevent System.MemberAccessException exceptions.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1LiveLinqInfo Class
C1LiveLinqInfo Members

C1.LiveLinq Namespace

Overview

Classes

	Class	Description
	LiveViewExtensions	Provides a set of static (extension) methods used in queries to define live views.
	SourceChangeEventArgs<T>	Provides data for the IObservableSource<T>.Changed event.

Interfaces

	Interface	Description
	IObservableSource<T>	Provides methods and events that are required for LiveLinq functionality, live views.
	ITransaction	Represents a transaction with an explicit scope.

Enumerations

	Enumeration	Description
	Order	Indicates if a certain order is required in the result collection of an operation.
	SourceChangeType	Describes a change occurring in a collection.
	TransactionState	Enumeration of the possible states an ITransaction can be in.

See Also

Reference

C1.Silverlight.LiveLinq Assembly

Classes

LiveViewExtensions
Provides a set of static (extension) methods used in queries to define live views.

Object Model

Syntax

Visual Basic (Declaration)
`Public MustInherit NotInheritable Class LiveViewExtensions`
C#
`public static class LiveViewExtensions`

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViewExtensions

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Members
C1.LiveLinq Namespace

Overview
Provides a set of static (extension) methods used in queries to define live views.

Object Model

Syntax

Visual Basic (Declaration)
`Public MustInherit NotInheritable Class LiveViewExtensions`
C#
`public static class LiveViewExtensions`

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViewExtensions

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Members
C1.LiveLinq Namespace

Members
Methods

The following tables list the members exposed by LiveViewExtensions.

Public Methods

	Name	Description
S	AsLive	Overloaded. Creates a view based on the specified IObservableSource<T> collection.
S	AsNonUpdatable<T>	Specifies a view as read-only.

S	AsUpdatable<T>	Specifies a view as updatable.
S	LiveAggregate	Overloaded. Applies an accumulator function over a view.
S	LiveAverage	Overloaded. Computes the average of a view of System.Int32 values.
S	LiveCount	Overloaded. A view representing the number of elements in a view.
S	LiveMax	Overloaded. Computes the maximum value of a view of System.Double values.
S	LiveMin	Overloaded. Computes the minimum value of a view of nullable System.Double values that are obtained by invoking a transform function on each element of the source view.
S	LiveSum	Overloaded. Computes the sum of a view of System.Int32 values.

Top

See Also

Reference

LiveViewExtensions Class
C1.LiveLinq Namespace

Methods
For a list of all members of this type, see LiveViewExtensions members.

Public Methods

	Name	Description
S	AsLive	Overloaded. Creates a view based on the specified IObservableSource<T> collection.
S	AsNonUpdatable<T>	Specifies a view as read-only.

S	AsUpdatable<T>	Specifies a view as updatable.
S	LiveAggregate	Overloaded. Applies an accumulator function over a view.
S	LiveAverage	Overloaded. Computes the average of a view of System.Int32 values.
S	LiveCount	Overloaded. A view representing the number of elements in a view.
S	LiveMax	Overloaded. Computes the maximum value of a view of System.Double values.
S	LiveMin	Overloaded. Computes the minimum value of a view of nullable System.Double values that are obtained by invoking a transform function on each element of the source view.
S	LiveSum	Overloaded. Computes the sum of a view of System.Int32 values.

Top

See Also

Reference

LiveViewExtensions Class
C1.LiveLinq Namespace

AsLive Method
Creates a view based on the specified IObservableSource<T> collection.

Overload List

Overload	Description
AsLive<T>(IObservableSource<T>)	Creates a view based on the specified IObservableSource<T> collection.
AsLive<T>(IObservableSource<T>,ViewOrder)	Creates a view based on the specified IObservableSource<T> collection.

AsLive<T>(INotifyCollectionChanged)	Creates a view based on the specified System.Collections.Specialized.INotifyCollectionChanged data source.
AsLive<T>(INotifyCollectionChanged,ViewOrder)	Creates a view based on the specified System.Collections.Specialized.INotifyCollectionChanged data source.
AsLive<T>(ObservableCollection<T>)	A typed specialization of the AsLive<T>(INotifyCollectionChanged) method.
AsLive<T>(ObservableCollection<T>,ViewOrder)	A typed specialization of the AsLive<T>(INotifyCollectionChanged,ViewOrder) method.
AsLive<T>(ReadOnlyObservableCollection<T>)	A typed specialization of the AsLive<T>(INotifyCollectionChanged) method.
AsLive<T>(ReadOnlyObservableCollection<T>,ViewOrder)	A typed specialization of the AsLive<T>(INotifyCollectionChanged,ViewOrder) method.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members

AsLive<T>(IObservableSource<T>) Method
The type of the elements in the collection.

The IObservableSource<T> collection to expose as a view.

Creates a view based on the specified IObservableSource<T> collection.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function AsLive(Of T)( _
   ByVal source As IObservableSource(Of T) _
) As View(Of T)
```

C#

```
public static View<T> AsLive<T>(
   IObservableSource<T> source
)
```

Parameters

source

The IObservableSource<T> collection to expose as a view.

Type Parameters

T

The type of the elements in the collection.

Return Value

A view that contains the same elements as the IObservableSource<T>

Remarks

The resulting view may have its elements ordered differently than they are ordered in the *source* collection. Correspondingly, views built on this resulting view (for example, if you filter it with **Where**) will not preserve the source order either. If you need to preserve the source order, consider using the other **AsLive** overload where you can specify to what extent you need the order to be preserved.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

AsLive<T>(IObservableSource<T>,ViewOrder) Method
The type of the elements in the collection.

The IObservableSource<T> collection to expose as a view.

Specifies whether to preserve source item order.

Creates a view based on the specified IObservableSource<T> collection.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function AsLive(Of T)( _
   ByVal source As IObservableSource(Of T), _
   ByVal order As ViewOrder _
) As View(Of T)
```

C#

```
public static View<T> AsLive<T>(
   IObservableSource<T> source,
   ViewOrder order
)
```

Parameters

source

The IObservableSource<T> collection to expose as a view.

order

Specifies whether to preserve source item order.

Type Parameters

T

The type of the elements in the collection.

Return Value

A view that contains the same elements as the IObservableSource<T>

Remarks

If the *order* parameter specifies preserving item order, the order of items in the source is preserved, at a certain performance cost, in the resulting view and in views based on it (for example, if you filter it with **Where**).

Note that **Join** does not preserve source order. If you need to order a join result, use **OrderBy** after **Join**.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

AsLive<T>(INotifyCollectionChanged) Method
The type of the elements in the view.

The System.Collections.Specialized.INotifyCollectionChanged data source to expose as a view.

Creates a view based on the specified System.Collections.Specialized.INotifyCollectionChanged data source.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function AsLive(Of T)( _
   ByVal source As INotifyCollectionChanged _
) As View(Of T)
```

C#

```
public static View<T> AsLive<T>(
   INotifyCollectionChanged source
)
```

Parameters

source

The System.Collections.Specialized.INotifyCollectionChanged data source to expose as a view.

Type Parameters

T

The type of the elements in the view.

Return Value

A view that contains the same elements as the System.Collections.Specialized.INotifyCollectionChanged data source.

Remarks

Use this method to build views from existing data sources implementing System.Collections.Specialized.INotifyCollectionChanged. The element type of this data source must implement System.ComponentModel.INotifyPropertyChanged, see Using the built-in collection class IndexedCollection(T) (LiveLinq to Objects)|tag=Using_the_built_in_collection_class_IndexedCollectionT_LiveLinq_to_Objects .

The resulting view may have its elements ordered differently than they are ordered in the *source*. Correspondingly, views built on this resulting view (for example, if you filter it with **Where**) will not preserve the source order either. If you need to preserve the source order, consider using the other **AsLive** overload where you can specify to what extent you need the order to be preserved.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

AsLive<T>(INotifyCollectionChanged,ViewOrder) Method
The type of the elements in the view.

The System.Collections.Specialized.INotifyCollectionChanged data source to expose as a view.

Specifies whether to preserve source item order.

Creates a view based on the specified System.Collections.Specialized.INotifyCollectionChanged data source.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function AsLive(Of T)( _
   ByVal source As INotifyCollectionChanged, _
   ByVal order As ViewOrder _
) As View(Of T)
```

C#

```
public static View<T> AsLive<T>(
   INotifyCollectionChanged source,
   ViewOrder order
)
```

Parameters

source

The System.Collections.Specialized.INotifyCollectionChanged data source to expose as a view.

order

Specifies whether to preserve source item order.

Type Parameters

T

The type of the elements in the view.

Return Value

A view that contains the same elements as the System.Collections.Specialized.INotifyCollectionChanged data source.

Remarks

Use this method to build views from existing data sources implementing System.Collections.Specialized.INotifyCollectionChanged. The element type of this data source must implement System.ComponentModel.INotifyPropertyChanged, see Using the built-in collection class IndexedCollection(T) (LiveLinq to Objects)|tag=Using_the_built_in_collection_class_IndexedCollectionT_LiveLinq_to_Objects.

If the *order* parameter specifies preserving item order, the order of items in the source is preserved, at a certain performance cost, in the resulting view and in views based on it (for example, if you filter it with **Where**).

Note that **Join** does not preserve source order. If you need to order a join result, use **OrderBy** after **Join**.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

AsLive<T>(ObservableCollection<T>) Method
The type of the elements in the view.

The ObservableCollection to expose as a view.

A typed specialization of the AsLive<T>(INotifyCollectionChanged) method.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function AsLive(Of T)( _
   ByVal source As ObservableCollection(Of T) _
) As View(Of T)
```

C#

```
public static View<T> AsLive<T>(
   ObservableCollection<T> source
)
```

Parameters

source

The ObservableCollection to expose as a view.

Type Parameters

T

The type of the elements in the view.

Return Value

A view that contains the same elements as the ObservableCollection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

AsLive<T>(ObservableCollection<T>,ViewOrder) Method
The type of the elements in the view.

The ObservableCollection to expose as a view.

Specifies whether to preserve source item order.

A typed specialization of the AsLive<T>(INotifyCollectionChanged,ViewOrder) method.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function AsLive(Of T)( _`
`  ByVal source As ObservableCollection(Of T), _`
`  ByVal order As ViewOrder _`
`) As View(Of T)`

C#

`public static View<T> AsLive<T>(`
`  ObservableCollection<T> source,`
`  ViewOrder order`
`)`

Parameters

source

The ObservableCollection to expose as a view.

order

Specifies whether to preserve source item order.

Type Parameters

T

The type of the elements in the view.

Return Value

A view that contains the same elements as the ObservableCollection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

AsLive<T>(ReadOnlyObservableCollection<T>) Method
The type of the elements in the view.

The ReadOnlyObservableCollection to expose as a view.

A typed specialization of the AsLive<T>(INotifyCollectionChanged) method.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function AsLive(Of T)( _`
`  ByVal source As ReadOnlyObservableCollection(Of T) _`
`) As View(Of T)`

C#

`public static View<T> AsLive<T>(`
`  ReadOnlyObservableCollection<T> source`
`)`

Parameters

source

The ReadOnlyObservableCollection to expose as a view.

Type Parameters

T

The type of the elements in the view.

Return Value

A view that contains the same elements as the ReadOnlyObservableCollection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

AsLive<T>(ReadOnlyObservableCollection<T>,ViewOrder) Method
The type of the elements in the view.

The ReadOnlyObservableCollection to expose as a view.

Specifies whether to preserve source item order.

A typed specialization of the AsLive<T>(INotifyCollectionChanged,ViewOrder) method.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function AsLive(Of T)( _
   ByVal source As ReadOnlyObservableCollection(Of T), _
   ByVal order As ViewOrder _
) As View(Of T)
```

C#

```
public static View<T> AsLive<T>(
   ReadOnlyObservableCollection<T> source,
   ViewOrder order
)
```

Parameters

source

The ReadOnlyObservableCollection to expose as a view.

order

Specifies whether to preserve source item order.

Type Parameters

T

The type of the elements in the view.

Return Value

A view that contains the same elements as the ReadOnlyObservableCollection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

AsNonUpdatable<T> Method
The type of the elements in the view.

The view to specify as read-only.

Specifies a view as read-only.

Syntax

Visual Basic (Declaration)

```
Public Shared Function AsNonUpdatable(Of T)( _
   ByVal view As View(Of T) _
) As View(Of T)
```

C#

```
public static View<T> AsNonUpdatable<T>(
   View<T> view
)
```

Parameters

view

The view to specify as read-only.

Type Parameters

T

The type of the elements in the view.

Return Value

The same *view*.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
AsUpdatable<T> Method

AsUpdatable<T> Method
The type of the elements in the view.

The view to specify as updatable.

Specifies a view as updatable.

Syntax

Visual Basic (Declaration)

```
Public Shared Function AsUpdatable(Of T)( _
   ByVal view As View(Of T) _
) As View(Of T)
```

C#

```
public static View<T> AsUpdatable<T>(
   View<T> view
)
```

Parameters

view

The view to specify as updatable.

Type Parameters

T

The type of the elements in the view.

Return Value

The same *view*.

Remarks

This method is used with **Join** operation to specify which of the two parts of the join you want to be updatable.

Properties exposed by a view can be updatable or read-only. Updatable properties of a view can be modified directly in the view. Read-only properties of a view cannot be modified directly in the view, but they still reflect up-to-date values of the source, so the difference is often not critical, you can always modify corresponding property in the source, that will automatically change the property in the view.

Only one of the two arguments of a **Join** can be updatable, the one you qualified with **AsUpdatable()**. In absence of this qualification, both parts of the join are read-only. For example, in `from c in customers.AsLive() join o in orders.AsLive().AsUpdatable() on o.CustomerID equals c.CustomerID select new { c.CustomerName, o.OrderDate, o.OrderAmount }` **CustomerName** is read-only, and **OrderDate** and **OrderAmount** are updatable.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
IsReadOnly Property
ViewRow Class
ViewRowState Enumeration

LiveAggregate Method
Applies an accumulator function over a view.

Overload List

Overload	Description
LiveAggregate<TSource>(View<TSource>,Expression<Func<TSource,TSource,TSource>>,Expression<Func<TSource,TSource,TSource>>,Expression<Func<TSource,TSource,Boolean>>)	Applies an accumulator function over a view.

LiveAggregate<TSource,TAccumulate>(View<TSource>,TAccumulate,Expression<Func<TAccumulate,TSource,TAccumulate>>,Expression<Func<TAccumulate,TSource,Boolean>>)	Applies an accumulator function over a view. The specified seed value is used as the initial accumulator value.

LiveAggregate<TSource,TAccumulate,TResult>(View<TSource>,TAccumulate,Expression<Func<TAccumulate,TSource,TAccumulate>>,Expression<Func<TAccumulate,TSource,TAccumulate>>,Expression<Func<TAccumulate,TSource,Boolean>>,Expression<Func<TAccumulate,TResult>>)	Applies an accumulator function over a view. The specified seed value is used as the initial accumulator value, and

	the specified function is used to select the result value.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members

LiveAggregate<TSource>(View<TSource>,Expression<Func<TSource,TSource,TSource>>,Expression<Func<TSource,TSource,TSource>>,Expression<Func<TSource,TSource,Boolean>>) Method
The type of the elements of *source*.

A view to aggregate over.

An accumulator function to be invoked on each element that is added to the source view.

A function to be applied to the accumulated value and to an element to obtain the changed accumulated value, when an element is removed from the source view.

A function used to determine whether *funcRemove* must be applied when an element is removed from the source view, or the accumulated value is not affected by its removal.

Applies an accumulator function over a view.

Syntax

Visual Basic (Declaration)	

```
Public Overloads Shared Function LiveAggregate(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal funcAdd As Expression(Of Func(Of TSource,TSource,TSource)), _
   ByVal funcRemove As Expression(Of Func(Of TSource,TSource,TSource)), _
   ByVal funcRemoveDefined As Expression(Of Func(Of TSource,TSource,Boolean)) _
) As AggregationView(Of TSource,TSource)
```

C#	

```
public static AggregationView<TSource,TSource> LiveAggregate<TSource>(
   View<TSource> source,
   Expression<Func<TSource,TSource,TSource>> funcAdd,
   Expression<Func<TSource,TSource,TSource>> funcRemove,
   Expression<Func<TSource,TSource,bool>> funcRemoveDefined
)
```

Parameters

source

A view to aggregate over.

funcAdd

An accumulator function to be invoked on each element that is added to the source view.

funcRemove

A function to be applied to the accumulated value and to an element to obtain the changed accumulated value, when an element is removed from the source view.

funcRemoveDefined

A function used to determine whether *funcRemove* must be applied when an element is removed from the source view, or the accumulated value is not affected by its removal.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the final accumulator value.

Remarks

It is possible to use standard LINQ query operator **Aggregate** instead of **LiveAggregate**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Aggregate** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAggregate** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAggregate<TSource,TAccumulate>(View<TSource>,TAccumulate,Expression<Func<TAccumulate,TSource,TAccumulate>>,Expression<Func<TAccumulate,TSource,TAccumulate>>,Expression<Func<TAccumulate,TSource,Boolean>>) Method

The type of the elements of *source*.

The type of the accumulator value.

A view to aggregate over.

The initial accumulator value.

An accumulator function to be invoked on each element that is added to the source view.

A function to be applied to the accumulated value and to an element to obtain the changed accumulated value, when an element is removed from the source view.

A function used to determine whether *funcRemove* must be applied when an element is removed from the source view, or the accumulated value is not affected by its removal.

Applies an accumulator function over a view. The specified seed value is used as the initial accumulator value.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveAggregate
    (Of TSource,TAccumulate)( _
   ByVal source As View(Of TSource), _
   ByVal seed As TAccumulate, _
   ByVal funcAdd As Expression(Of Func(Of TAccumulate,TSource,TAccumulate)), _
   ByVal funcRemove As Expression(Of Func(Of TAccumulate,TSource,TAccumulate)),
_
   ByVal funcRemoveDefined As Expression(Of Func(Of
TAccumulate,TSource,Boolean)) _
) As AggregationView(Of TSource,TAccumulate)
```

C#

```
public static AggregationView<TSource,TAccumulate>
LiveAggregate<TSource,TAccumulate>(
   View<TSource> source,
   TAccumulate seed,
   Expression<Func<TAccumulate,TSource,TAccumulate>> funcAdd,
   Expression<Func<TAccumulate,TSource,TAccumulate>> funcRemove,
   Expression<Func<TAccumulate,TSource,bool>> funcRemoveDefined
)
```

Parameters

source

A view to aggregate over.

seed

The initial accumulator value.

funcAdd

An accumulator function to be invoked on each element that is added to the source view.

funcRemove

A function to be applied to the accumulated value and to an element to obtain the changed accumulated value, when an element is removed from the source view.

funcRemoveDefined

A function used to determine whether *funcRemove* must be applied when an element is removed from the source view, or the accumulated value is not affected by its removal.

Type Parameters

TSource

The type of the elements of *source*.

TAccumulate

The type of the accumulator value.

Return Value

A view representing the final accumulator value.

Remarks

It is possible to use standard LINQ query operator **Aggregate** instead of **LiveAggregate**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Aggregate** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAggregate** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAggregate<TSource,TAccumulate,TResult>(View<TSource>,TAccumulate,Expression<Func<TAccumulate,TSource,TAccumulate>>,Expression<Func<TAccumulate,TSource,TAccumulate>>,Expression<Func<TAccumulate,TSource,Boolean>>,Expression<Func<TAccumulate,TResult>>) Method

The type of the elements of *source*.

The type of the accumulator value.

The type of the resulting value.

A view to aggregate over.

The initial accumulator value.

An accumulator function to be invoked on each element that is added to the source view.

A function to be applied to the accumulated value and to an element to obtain the changed accumulated value, when an element is removed from the source view.

A function used to determine whether *funcRemove* must be applied when an element is removed from the source view, or the accumulated value is not affected by its removal.

A function to transform the final accumulator value into the result value.

Applies an accumulator function over a view. The specified seed value is used as the initial accumulator value, and the specified function is used to select the result value.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveAggregate
    (Of TSource,TAccumulate,TResult)( _
   ByVal source As View(Of TSource), _
   ByVal seed As TAccumulate, _
   ByVal funcAdd As Expression(Of Func(Of TAccumulate,TSource,TAccumulate)), _
   ByVal funcRemove As Expression(Of Func(Of TAccumulate,TSource,TAccumulate)),
_
```

```
   ByVal funcRemoveDefined As Expression(Of Func(Of
TAccumulate,TSource,Boolean)), _
   ByVal resultSelector As Expression(Of Func(Of TAccumulate,TResult)) _
) As AggregationView(Of TSource,TResult)
```

C#

```
public static AggregationView<TSource,TResult>
LiveAggregate<TSource,TAccumulate,TResult>(
   View<TSource> source,
   TAccumulate seed,
   Expression<Func<TAccumulate,TSource,TAccumulate>> funcAdd,
   Expression<Func<TAccumulate,TSource,TAccumulate>> funcRemove,
   Expression<Func<TAccumulate,TSource,bool>> funcRemoveDefined,
   Expression<Func<TAccumulate,TResult>> resultSelector
)
```

Parameters

source

A view to aggregate over.

seed

The initial accumulator value.

funcAdd

An accumulator function to be invoked on each element that is added to the source view.

funcRemove

A function to be applied to the accumulated value and to an element to obtain the changed accumulated value, when an element is removed from the source view.

funcRemoveDefined

A function used to determine whether *funcRemove* must be applied when an element is removed from the source view, or the accumulated value is not affected by its removal.

resultSelector

A function to transform the final accumulator value into the result value.

Type Parameters

TSource

The type of the elements of *source*.

TAccumulate

The type of the accumulator value.

TResult

The type of the resulting value.

Return Value

A view representing the final accumulator value.

Remarks

It is possible to use standard LINQ query operator **Aggregate** instead of **LiveAggregate**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Aggregate** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAggregate** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage Method

Computes the average of a view of System.Int32 values.

Overload List

Overload	Description
LiveAverage(View<Int32>)	Computes the average of a view of System.Int32 values.
LiveAverage(View<Nullable<Int32>>)	Computes the average of a view of nullable System.Int32 values.
LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Int32>>)	Computes the average of a view of System.Int32 values that are obtained by invoking a transform function on each element of the source view.
LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Int32>>>)	Computes the average of a view of

	nullable System.Int32 values that are obtained by invoking a transform function on each element of the source view.
LiveAverage(View<Int64>)	Computes the average of a view of System.Int64 values.
LiveAverage(View<Nullable<Int64>>)	Computes the average of a view of nullable System.Int64 values.
LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Int64>>)	Computes the average of a view of System.Int64 values that are obtained by invoking a transform function on each element

	of the source view.
LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Int64>>>)	Computes the average of a view of nullable System.Int64 values that are obtained by invoking a transform function on each element of the source view.
LiveAverage(View<Decimal>)	Computes the average of a view of System.Decimal values.
LiveAverage(View<Nullable<Decimal>>)	Computes the average of a view of nullable System.Decimal values.
LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Decimal>>)	Computes the average of a view of System.Decim

	al values that are obtained by invoking a transform function on each element of the source view.
LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Decimal>>>)	Computes the average of a view of nullable System.Decimal values that are obtained by invoking a transform function on each element of the source view.
LiveAverage(View<Double>)	Computes the average of a view of System.Double values.
LiveAverage(View<Nullable<Double>>)	Computes the average of a view of nullable System.Doubl

	e values.
LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Double>>)	Computes the average of a view of System.Double values that are obtained by invoking a transform function on each element of the source view.
LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Double>>>)	Computes the average of a view of nullable System.Double values that are obtained by invoking a transform function on each element of the source view.
LiveAverage(View<Single>)	Computes the average of a view of System.Single values.

LiveAverage(View<Nullable<Single>>)	Computes the average of a view of nullable System.Single values.
LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Single>>)	Computes the average of a view of System.Single values that are obtained by invoking a transform function on each element of the source view.
LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Single>>>)	Computes the average of a view of nullable System.Single values that are obtained by invoking a transform function on each element of the source view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members

LiveAverage(View<Int32>) Method
A view containing the values to calculate the average of.

Computes the average of a view of System.Int32 values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveAverage( _
   ByVal source As View(Of Integer) _
) As AggregationView(Of Integer,Double)
```

C#

```
public static AggregationView<int,double> LiveAverage(
   View<int> source
)
```

Parameters

source

A view containing the values to calculate the average of.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty or contains only nulls, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the

source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage(View<Nullable<Int32>>) Method
A view containing the values to calculate the average of.

Computes the average of a view of nullable System.Int32 values.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function LiveAverage( _`
`  ByVal source As View(Of Nullable(Of Integer)) _`
`) As AggregationView(Of Nullable(Of Integer),Nullable(Of Double))`

C#

`public static AggregationView<Nullable<int>,Nullable<double>> LiveAverage(`
`  View<Nullable<int>> source`
`)`

Parameters

source

A view containing the values to calculate the average of.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty or contains only nulls, the average value is null.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Int32>>) Method
The type of the elements of *source*.

A view containing the values to calculate the average of.

A transform function to apply to each element.

Computes the average of a view of System.Int32 values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveAverage(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Integer)) _
```

```
) As AggregationView(Of TSource,Double)
```

C#

```
public static AggregationView<TSource,double> LiveAverage<TSource>(
   View<TSource> source,
   Expression<Func<TSource,int>> selector
)
```

Parameters

source

A view containing the values to calculate the average of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Int32>>>) Method
The type of the elements of *source*.

A view containing the values to calculate the average of.

A transform function to apply to each element.

Computes the average of a view of nullable System.Int32 values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveAverage(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Integer))) _
) As AggregationView(Of TSource,Nullable(Of Double))
```

C#

```
public static AggregationView<TSource,Nullable<double>> LiveAverage<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<int>>> selector
)
```

Parameters

source

A view containing the values to calculate the average of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty, the average value is null.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage(View<Int64>) Method
A view containing the values to calculate the average of.

Computes the average of a view of System.Int64 values.

Syntax

Visual Basic (Declaration)	

```
Public Overloads Shared Function LiveAverage( _
   ByVal source As View(Of Long) _
```

```
) As AggregationView(Of Long,Double)
```

C#

```
public static AggregationView<long,double> LiveAverage(
   View<long> source
)
```

Parameters

source

A view containing the values to calculate the average of.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty or contains only nulls, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage(View<Nullable<Int64>>) Method
A view containing the values to calculate the average of.

Computes the average of a view of nullable System.Int64 values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveAverage( _
   ByVal source As View(Of Nullable(Of Long)) _
) As AggregationView(Of Nullable(Of Long),Nullable(Of Double))
```

C#

```
public static AggregationView<Nullable<long>,Nullable<double>> LiveAverage(
   View<Nullable<long>> source
)
```

Parameters

source

A view containing the values to calculate the average of.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty or contains only nulls, the average value is null.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Int64>>) Method
The type of the elements of *source*.

A view containing the values to calculate the average of.

A transform function to apply to each element.

Computes the average of a view of System.Int64 values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function LiveAverage(Of TSource)( _`
`  ByVal source As View(Of TSource), _`
`  ByVal selector As Expression(Of Func(Of TSource,Long)) _`
`) As AggregationView(Of TSource,Double)`

C#

`public static AggregationView<TSource,double> LiveAverage<TSource>(`
`  View<TSource> source,`
`  Expression<Func<TSource,long>> selector`
`)`

Parameters

source

A view containing the values to calculate the average of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Int64>>>) Method
The type of the elements of *source*.

A view containing the values to calculate the average of.

A transform function to apply to each element.

Computes the average of a view of nullable System.Int64 values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)	
`Public Overloads Shared Function LiveAverage(Of TSource)( _`	

```
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Long))) _
) As AggregationView(Of TSource,Nullable(Of Double))
```

C#

```
public static AggregationView<TSource,Nullable<double>> LiveAverage<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<long>>> selector
)
```

Parameters

source

A view containing the values to calculate the average of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty, the average value is null.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage(View<Decimal>) Method
A view containing the values to calculate the average of.

Computes the average of a view of System.Decimal values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveAverage( _
   ByVal source As View(Of Decimal) _
) As AggregationView(Of Decimal,Decimal)
```

C#

```
public static AggregationView<decimal,decimal> LiveAverage(
   View<decimal> source
)
```

Parameters

source

A view containing the values to calculate the average of.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty or contains only nulls, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage(View<Nullable<Decimal>>) Method
A view containing the values to calculate the average of.

Computes the average of a view of nullable System.Decimal values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveAverage( _
   ByVal source As View(Of Nullable(Of Decimal)) _
) As AggregationView(Of Nullable(Of Decimal),Nullable(Of Decimal))
```

C#

```
public static AggregationView<Nullable<decimal>,Nullable<decimal>> LiveAverage(
   View<Nullable<decimal>> source
)
```

Parameters

source

A view containing the values to calculate the average of.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty or contains only nulls, the average value is null.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Decimal>>) Method
The type of the elements of *source*.

A view containing the values to calculate the average of.

A transform function to apply to each element.

Computes the average of a view of System.Decimal values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)	
`Public Overloads Shared Function LiveAverage(Of TSource)( _`	

```
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Decimal)) _
) As AggregationView(Of TSource,Decimal)
```

C#

```
public static AggregationView<TSource,decimal> LiveAverage<TSource>(
   View<TSource> source,
   Expression<Func<TSource,decimal>> selector
)
```

Parameters

source

A view containing the values to calculate the average of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Decimal>>>) Method
The type of the elements of *source*.

A view containing the values to calculate the average of.

A transform function to apply to each element.

Computes the average of a view of nullable System.Decimal values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveAverage(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Decimal))) _
) As AggregationView(Of TSource,Nullable(Of Decimal))
```

C#

```
public static AggregationView<TSource,Nullable<decimal>> LiveAverage<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<decimal>>> selector
)
```

Parameters

source

A view containing the values to calculate the average of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty, the average value is null.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage(View<Double>) Method
A view containing the values to calculate the average of.

Computes the average of a view of System.Double values.

Syntax

Visual Basic (Declaration)	

```
Public Overloads Shared Function LiveAverage( _
   ByVal source As View(Of Double) _
) As AggregationView(Of Double,Double)
```

C#

```
public static AggregationView<double,double> LiveAverage(
   View<double> source
)
```

Parameters

source

A view containing the values to calculate the average of.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty or contains only nulls, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage(View<Nullable<Double>>) Method

A view containing the values to calculate the average of.

Computes the average of a view of nullable System.Double values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveAverage( _
   ByVal source As View(Of Nullable(Of Double)) _
) As AggregationView(Of Nullable(Of Double),Nullable(Of Double))
```

C#

```
public static AggregationView<Nullable<double>,Nullable<double>> LiveAverage(
   View<Nullable<double>> source
)
```

Parameters

source

A view containing the values to calculate the average of.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty or contains only nulls, the average value is null.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Double>>) Method
The type of the elements of *source*.

A view containing the values to calculate the average of.

A transform function to apply to each element.

Computes the average of a view of System.Double values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveAverage(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Double)) _
) As AggregationView(Of TSource,Double)
```

C#

```
public static AggregationView<TSource,double> LiveAverage<TSource>(
   View<TSource> source,
   Expression<Func<TSource,double>> selector
)
```

Parameters

source

A view containing the values to calculate the average of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Double>>>) Method
The type of the elements of *source*.

A view containing the values to calculate the average of.

A transform function to apply to each element.

Computes the average of a view of nullable System.Double values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveAverage(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Double))) _
) As AggregationView(Of TSource,Nullable(Of Double))
```

C#

```
public static AggregationView<TSource,Nullable<double>> LiveAverage<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<double>>> selector
)
```

Parameters

source

A view containing the values to calculate the average of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty, the average value is null.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage(View<Single>) Method
A view containing the values to calculate the average of.

Computes the average of a view of System.Single values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveAverage( _
   ByVal source As View(Of Single) _
) As AggregationView(Of Single,Double)
```

C#

```
public static AggregationView<float,double> LiveAverage(
   View<float> source
)
```

Parameters

source

A view containing the values to calculate the average of.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty or contains only nulls, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage(View<Nullable<Single>>) Method
A view containing the values to calculate the average of.

Computes the average of a view of nullable System.Single values.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function LiveAverage( _`
`  ByVal source As View(Of Nullable(Of Single)) _`
`) As AggregationView(Of Nullable(Of Single),Nullable(Of Double))`

C#

`public static AggregationView<Nullable<float>,Nullable<double>> LiveAverage(`
`  View<Nullable<float>> source`
`)`

Parameters

source

A view containing the values to calculate the average of.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty or contains only nulls, the average value is null.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Single>>) Method
The type of the elements of *source*.

A view containing the values to calculate the average of.

A transform function to apply to each element.

Computes the average of a view of System.Single values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)
`Public Overloads Shared Function LiveAverage(Of TSource)( _`

```
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Single)) _
) As AggregationView(Of TSource,Double)
```

C#

```
public static AggregationView<TSource,double> LiveAverage<TSource>(
   View<TSource> source,
   Expression<Func<TSource,float>> selector
)
```

Parameters

source

A view containing the values to calculate the average of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveAverage<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Single>>>) Method
The type of the elements of *source*.

A view containing the values to calculate the average of.

A transform function to apply to each element.

Computes the average of a view of nullable System.Single values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveAverage(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Single))) _
) As AggregationView(Of TSource,Nullable(Of Double))
```

C#

```
public static AggregationView<TSource,Nullable<double>> LiveAverage<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<float>>> selector
)
```

Parameters

source

A view containing the values to calculate the average of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the average of the values.

Remarks

If the *source* is empty, the average value is null.

It is possible to use standard LINQ query operator **Average** instead of **LiveAverage**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Average** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveAverage** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveCount Method
A view representing the number of elements in a view.

Overload List

Overload	Description

LiveCount<T>(View<T>)	A view representing the number of elements in a view.
LiveCount<T>(View<T>,Expression<Func<T,Boolean>>)	A view representing the number of elements of the specified view satisfying a condition.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members

LiveCount<T>(View<T>) Method
The type of the elements of *source*.

A view that contains elements to be counted.

A view representing the number of elements in a view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveCount(Of T)( _
   ByVal source As View(Of T) _
) As AggregationView(Of T,Integer)
```

C#

```
public static AggregationView<T,int> LiveCount<T>(
   View<T> source
)
```

Parameters

source

A view that contains elements to be counted.

Type Parameters

T

The type of the elements of *source*.

Remarks

It is possible to use standard LINQ query operator **Count** instead of **LiveCount**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Count** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveCount** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveCount<T>(View<T>,Expression<Func<T,Boolean>>) Method
The type of the elements of *source*.

A view that contains elements to be tested and counted.

A function to test each element for a condition.

A view representing the number of elements of the specified view satisfying a condition.

Syntax

Visual Basic (Declaration)	

```
Public Overloads Shared Function LiveCount(Of T)( _
   ByVal source As View(Of T), _
   ByVal predicate As Expression(Of Func(Of T,Boolean)) _
) As AggregationView(Of T,Integer)
```

C#

```
public static AggregationView<T,int> LiveCount<T>(
   View<T> source,
   Expression<Func<T,bool>> predicate
)
```

Parameters

source

A view that contains elements to be tested and counted.

predicate

A function to test each element for a condition.

Type Parameters

T

The type of the elements of *source*.

Remarks

It is possible to use standard LINQ query operator **Count** instead of **LiveCount**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Count** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveCount** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax Method
Computes the maximum value of a view of System.Double values.

Overload List

Overload	Description
LiveMax(View<Double>)	Computes the maximum value of a view of System.Double values.
LiveMax(View<Nullable<Double>>)	Computes the maximum value of a view of nullable System.Double values.
LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Double>>)	Computes the maximum value of a view of System.Double values that are obtained by invoking a transform function on

	each element of the source view.
LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Double>>>)	Computes the maximum value of a view of nullable System.Double values that are obtained by invoking a transform function on each element of the source view.
LiveMax(View<Single>)	Computes the maximum value of a view of System.Single values.
LiveMax(View<Nullable<Single>>)	Computes the maximum value of a view of nullable System.Single values.
LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Single>>)	Computes the maximum

	value of a view of System.Single values that are obtained by invoking a transform function on each element of the source view.
LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Single>>>)	Computes the maximum value of a view of nullable System.Single values that are obtained by invoking a transform function on each element of the source view.
LiveMax<TSource,TResult>(View<TSource>,Expression<Func<TSource,TResult>>)	Invokes a transform function on each element of a view of elements of a generic type and computes

	the maximum resulting value.
LiveMax<TSource>(View<TSource>)	Computes the maximum value of a view of elements of a generic type.
LiveMax(View<Int32>)	Computes the maximum value of a view of System.Int32 values.
LiveMax(View<Nullable<Int32>>)	Computes the maximum value of a view of nullable System.Int32 values.
LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Int32>>)	Computes the maximum value of a view of System.Int32 values that are obtained by invoking a transform

	function on each element of the source view.
LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Int32>>>)	Computes the maximum value of a view of nullable System.Int32 values that are obtained by invoking a transform function on each element of the source view.
LiveMax(View<Decimal>)	Computes the maximum value of a view of System.Decimal values.
LiveMax(View<Nullable<Decimal>>)	Computes the maximum value of a view of nullable System.Decimal values.
LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Decimal>>)	Computes the

	maximum value of a view of System.Decimal values that are obtained by invoking a transform function on each element of the source view.
LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Decimal>>>)	Computes the maximum value of a view of nullable System.Decimal values that are obtained by invoking a transform function on each element of the source view.
LiveMax(View<Int64>)	Computes the maximum value of a view of System.Int64 values.

LiveMax(View<Nullable<Int64>>)	Computes the maximum value of a view of nullable System.Int64 values.
LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Int64>>)	Computes the maximum value of a view of System.Int64 values that are obtained by invoking a transform function on each element of the source view.
LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Int64>>>)	Computes the maximum value of a view of nullable System.Int64 values that are obtained by invoking a transform function on each element of the source view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members

LiveMax(View<Double>) Method
A view containing the values to determine the maximum value of.

Computes the maximum value of a view of System.Double values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMax( _
   ByVal source As View(Of Double) _
) As AggregationView(Of Double,Double)
```

C#

```
public static AggregationView<double,double> LiveMax(
   View<double> source
)
```

Parameters

source

A view containing the values to determine the maximum value of.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty or contains only nulls, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax(View<Nullable<Double>>) Method
A view containing the values to determine the maximum value of.

Computes the maximum value of a view of nullable System.Double values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMax( _
   ByVal source As View(Of Nullable(Of Double)) _
) As AggregationView(Of Nullable(Of Double),Nullable(Of Double))
```

C#

```
public static AggregationView<Nullable<double>,Nullable<double>> LiveMax(
   View<Nullable<double>> source
)
```

Parameters

source

A view containing the values to determine the maximum value of.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty or contains only nulls, the maximum value is null.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Double>>) Method
The type of the elements of *source*.

A view containing the values to determine the maximum value of.

A transform function to apply to each element.

Computes the maximum value of a view of System.Double values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)	
`Public Overloads Shared Function LiveMax(Of TSource)( _` `  ByVal source As View(Of TSource), _`	

```
   ByVal selector As Expression(Of Func(Of TSource,Double)) _
) As AggregationView(Of TSource,Double)
```

C#

```
public static AggregationView<TSource,double> LiveMax<TSource>(
   View<TSource> source,
   Expression<Func<TSource,double>> selector
)
```

Parameters

source

A view containing the values to determine the maximum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Double>>>) Method
The type of the elements of *source*.

A view containing the values to determine the maximum value of.

A transform function to apply to each element.

Computes the maximum value of a view of nullable System.Double values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMax(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Double))) _
) As AggregationView(Of TSource,Nullable(Of Double))
```

C#

```
public static AggregationView<TSource,Nullable<double>> LiveMax<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<double>>> selector
)
```

Parameters

source

A view containing the values to determine the maximum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty or contains only nulls, the maximum value is null.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax(View<Single>) Method
A view containing the values to determine the maximum value of.

Computes the maximum value of a view of System.Single values.

Syntax

Visual Basic (Declaration)	

```
Public Overloads Shared Function LiveMax( _
   ByVal source As View(Of Single) _
```

```
) As AggregationView(Of Single,Single)
```

C#

```
public static AggregationView<float,float> LiveMax(
   View<float> source
)
```

Parameters

source

A view containing the values to determine the maximum value of.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty or contains only nulls, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax(View<Nullable<Single>>) Method
A view containing the values to determine the maximum value of.

Computes the maximum value of a view of nullable System.Single values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMax( _
   ByVal source As View(Of Nullable(Of Single)) _
) As AggregationView(Of Nullable(Of Single),Nullable(Of Single))
```

C#

```
public static AggregationView<Nullable<float>,Nullable<float>> LiveMax(
   View<Nullable<float>> source
)
```

Parameters

source

A view containing the values to determine the maximum value of.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty or contains only nulls, the maximum value is null.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Single>>) Method
The type of the elements of *source*.

A view containing the values to determine the maximum value of.

A transform function to apply to each element.

Computes the maximum value of a view of System.Single values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMax(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Single)) _
) As AggregationView(Of TSource,Single)
```

C#

```
public static AggregationView<TSource,float> LiveMax<TSource>(
   View<TSource> source,
   Expression<Func<TSource,float>> selector
)
```

Parameters

source

A view containing the values to determine the maximum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Single>>>) Method
The type of the elements of *source*.

A view containing the values to determine the maximum value of.

A transform function to apply to each element.

Computes the maximum value of a view of nullable System.Single values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMax(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Single))) _
```

```
) As AggregationView(Of TSource,Nullable(Of Single))
```

C#

```
public static AggregationView<TSource,Nullable<float>> LiveMax<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<float>>> selector
)
```

Parameters

source

A view containing the values to determine the maximum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty or contains only nulls, the maximum value is null.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax<TSource,TResult>(View<TSource>,Expression<Func<TSource,TResult>>) Method
The type of the elements of *source*.

The type of the value returned by *selector*.

A view containing the values to determine the maximum value of.

A transform function to apply to each element.

Invokes a transform function on each element of a view of elements of a generic type and computes the maximum resulting value.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMax
    (Of TSource,TResult)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,TResult)) _
) As AggregationView(Of TSource,TResult)
```

C#

```
public static AggregationView<TSource,TResult> LiveMax<TSource,TResult>(
   View<TSource> source,
   Expression<Func<TSource,TResult>> selector
)
```

Parameters

source

A view containing the values to determine the maximum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

TResult

The type of the value returned by *selector*.

Return Value

A view representing the maximum of the values.

Remarks

If type *TResult* implements IComparable, this method uses that implementation to compare values. Otherwise, if type *TResult* implements System.IComparable, that implementation is used to compare values.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax<TSource>(View<TSource>) Method
The type of the elements of *source*.

A view containing the values to determine the maximum value of.

Computes the maximum value of a view of elements of a generic type.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMax(Of TSource)( _
   ByVal source As View(Of TSource) _
) As AggregationView(Of TSource,TSource)
```

C#

```
public static AggregationView<TSource,TSource> LiveMax<TSource>(
   View<TSource> source
)
```

Parameters

source

A view containing the values to determine the maximum value of.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the maximum of the values.

Remarks

If type *TSource* implements IComparable, this method uses that implementation to compare values. Otherwise, if type *TSource* implements System.IComparable, that implementation is used to compare values.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax(View<Int32>) Method
A view containing the values to determine the maximum value of.

Computes the maximum value of a view of System.Int32 values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMax( _
   ByVal source As View(Of Integer) _
) As AggregationView(Of Integer,Integer)
```

C#

```
public static AggregationView<int,int> LiveMax(
   View<int> source
)
```

Parameters

source

A view containing the values to determine the maximum value of.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty or contains only nulls, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax(View<Nullable<Int32>>) Method
A view containing the values to determine the maximum value of.

Computes the maximum value of a view of nullable System.Int32 values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMax( _
   ByVal source As View(Of Nullable(Of Integer)) _
) As AggregationView(Of Nullable(Of Integer),Nullable(Of Integer))
```

C#

```
public static AggregationView<Nullable<int>,Nullable<int>> LiveMax(
   View<Nullable<int>> source
)
```

Parameters

source

A view containing the values to determine the maximum value of.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty or contains only nulls, the maximum value is null.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Int32>>) Method
The type of the elements of *source*.

A view containing the values to determine the maximum value of.

A transform function to apply to each element.

Computes the maximum value of a view of System.Int32 values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)	
`Public Overloads Shared Function LiveMax(Of TSource)( _`	

```
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Integer)) _
) As AggregationView(Of TSource,Integer)
```

C#

```
public static AggregationView<TSource,int> LiveMax<TSource>(
   View<TSource> source,
   Expression<Func<TSource,int>> selector
)
```

Parameters

source

A view containing the values to determine the maximum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Int32>>>) Method
The type of the elements of *source*.

A view containing the values to determine the maximum value of.

A transform function to apply to each element.

Computes the maximum value of a view of nullable System.Int32 values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMax(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Integer))) _
) As AggregationView(Of TSource,Nullable(Of Integer))
```

C#

```
public static AggregationView<TSource,Nullable<int>> LiveMax<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<int>>> selector
)
```

Parameters

source

A view containing the values to determine the maximum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty or contains only nulls, the maximum value is null.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax(View<Decimal>) Method
A view containing the values to determine the maximum value of.

Computes the maximum value of a view of System.Decimal values.

Syntax

Visual Basic (Declaration)	

```
Public Overloads Shared Function LiveMax( _
   ByVal source As View(Of Decimal) _
) As AggregationView(Of Decimal,Decimal)
```

C#

```
public static AggregationView<decimal,decimal> LiveMax(
   View<decimal> source
)
```

Parameters

source

A view containing the values to determine the maximum value of.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty or contains only nulls, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax(View<Nullable<Decimal>>) Method
A view containing the values to determine the maximum value of.

Computes the maximum value of a view of nullable System.Decimal values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMax( _
   ByVal source As View(Of Nullable(Of Decimal)) _
) As AggregationView(Of Nullable(Of Decimal),Nullable(Of Decimal))
```

C#

```
public static AggregationView<Nullable<decimal>,Nullable<decimal>> LiveMax(
   View<Nullable<decimal>> source
)
```

Parameters

source

A view containing the values to determine the maximum value of.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty or contains only nulls, the maximum value is null.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Decimal>>) Method
The type of the elements of *source*.

A view containing the values to determine the maximum value of.

A transform function to apply to each element.

Computes the maximum value of a view of System.Decimal values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMax(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Decimal)) _
) As AggregationView(Of TSource,Decimal)
```

C#

```
public static AggregationView<TSource,decimal> LiveMax<TSource>(
   View<TSource> source,
   Expression<Func<TSource,decimal>> selector
)
```

Parameters

source

A view containing the values to determine the maximum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Decimal>>>) Method
The type of the elements of *source*.

A view containing the values to determine the maximum value of.

A transform function to apply to each element.

Computes the maximum value of a view of nullable System.Decimal values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMax(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Decimal))) _
```

```
) As AggregationView(Of TSource,Nullable(Of Decimal))
```

C#

```
public static AggregationView<TSource,Nullable<decimal>> LiveMax<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<decimal>>> selector
)
```

Parameters

source

A view containing the values to determine the maximum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty or contains only nulls, the maximum value is null.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax(View<Int64>) Method

A view containing the values to determine the maximum value of.

Computes the maximum value of a view of System.Int64 values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMax( _
   ByVal source As View(Of Long) _
) As AggregationView(Of Long,Long)
```

C#

```
public static AggregationView<long,long> LiveMax(
   View<long> source
)
```

Parameters

source

A view containing the values to determine the maximum value of.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty or contains only nulls, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate

it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax(View<Nullable<Int64>>) Method
A view containing the values to determine the maximum value of.

Computes the maximum value of a view of nullable System.Int64 values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMax( _
   ByVal source As View(Of Nullable(Of Long)) _
) As AggregationView(Of Nullable(Of Long),Nullable(Of Long))
```

C#

```
public static AggregationView<Nullable<long>,Nullable<long>> LiveMax(
   View<Nullable<long>> source
)
```

Parameters

source

A view containing the values to determine the maximum value of.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty or contains only nulls, the maximum value is null.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Int64>>) Method
The type of the elements of *source*.

A view containing the values to determine the maximum value of.

A transform function to apply to each element.

Computes the maximum value of a view of System.Int64 values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMax(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Long)) _
) As AggregationView(Of TSource,Long)
```

C#

```
public static AggregationView<TSource,long> LiveMax<TSource>(
   View<TSource> source,
   Expression<Func<TSource,long>> selector
)
```

Parameters

source

A view containing the values to determine the maximum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMax<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Int64>>>) Method
The type of the elements of *source*.

A view containing the values to determine the maximum value of.

A transform function to apply to each element.

Computes the maximum value of a view of nullable System.Int64 values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMax(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Long))) _
) As AggregationView(Of TSource,Nullable(Of Long))
```

C#

```
public static AggregationView<TSource,Nullable<long>> LiveMax<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<long>>> selector
)
```

Parameters

source

A view containing the values to determine the maximum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the maximum of the values.

Remarks

If the *source* is empty or contains only nulls, the maximum value is null.

It is possible to use standard LINQ query operator **Max** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Max** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin Method
Computes the minimum value of a view of nullable System.Double values that are obtained by invoking a transform function on each element of the source view.

Overload List

Overload	Description
LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Double>>>)	Computes the minimum value of a view of nullable System.Double

	values that are obtained by invoking a transform function on each element of the source view.
LiveMin(View<Single>)	Computes the minimum value of a view of System.Single values.
LiveMin(View<Nullable<Single>>)	Computes the minimum value of a view of nullable System.Single values.
LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Single>>)	Computes the minimum value of a view of System.Single values that are obtained by invoking a transform function on each element

	of the source view.
LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Single>>>)	Computes the minimum value of a view of nullable System.Single values that are obtained by invoking a transform function on each element of the source view.
LiveMin<TSource,TResult>(View<TSource>,Expression<Func<TSource,TResult>>)	Invokes a transform function on each element of a view of elements of a generic type and computes the minimum resulting value.
LiveMin<TSource>(View<TSource>)	Computes the minimum value of a view of elements of a generic

	type.
LiveMin(View<Int32>)	Computes the minimum value of a view of System.Int32 values.
LiveMin(View<Nullable<Int32>>)	Computes the minimum value of a view of nullable System.Int32 values.
LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Int32>>)	Computes the minimum value of a view of System.Int32 values that are obtained by invoking a transform function on each element of the source view.
LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Int32>>>)	Computes the minimum value of a view of nullable

	System.Int32 values that are obtained by invoking a transform function on each element of the source view.
LiveMin(View<Decimal>)	Computes the minimum value of a view of System.Decimal values.
LiveMin(View<Nullable<Decimal>>)	Computes the minimum value of a view of nullable System.Decimal values.
LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Decimal>>)	Computes the minimum value of a view of System.Decimal values that are obtained by invoking a transform function on

	each element of the source view.
LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Decimal>>>)	Computes the minimum value of a view of nullable System.Decimal values that are obtained by invoking a transform function on each element of the source view.
LiveMin(View<Int64>)	Computes the minimum value of a view of System.Int64 values.
LiveMin(View<Nullable<Int64>>)	Computes the minimum value of a view of nullable System.Int64 values.
LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Int64>>)	Computes the minimum

	value of a view of System.Int64 values that are obtained by invoking a transform function on each element of the source view.
LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Int64>>>)	Computes the minimum value of a view of nullable System.Int64 values that are obtained by invoking a transform function on each element of the source view.
LiveMin(View<Double>)	Computes the minimum value of a view of System.Double values.
LiveMin(View<Nullable<Double>>)	Computes the

	minimum value of a view of nullable System.Double values.
LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Double>>)	Computes the minimum value of a view of System.Double values that are obtained by invoking a transform function on each element of the source view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members

LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Double>>>) Method
The type of the elements of *source*.

A view containing the values to determine the minimum value of.

A transform function to apply to each element.

Computes the minimum value of a view of nullable System.Double values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMin(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Double))) _
) As AggregationView(Of TSource,Nullable(Of Double))
```

C#

```
public static AggregationView<TSource,Nullable<double>> LiveMin<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<double>>> selector
)
```

Parameters

source

A view containing the values to determine the minimum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty or contains only nulls, the minimum value is null.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the

entire source collection and aggregate it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin(View<Single>) Method
A view containing the values to determine the minimum value of.

Computes the minimum value of a view of System.Single values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMin( _
   ByVal source As View(Of Single) _
) As AggregationView(Of Single,Single)
```

C#

```
public static AggregationView<float,float> LiveMin(
   View<float> source
)
```

Parameters

source

A view containing the values to determine the minimum value of.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty or contains only nulls, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin(View<Nullable<Single>>) Method
A view containing the values to determine the minimum value of.

Computes the minimum value of a view of nullable System.Single values.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function LiveMin( _`
`  ByVal source As View(Of Nullable(Of Single)) _`
`) As AggregationView(Of Nullable(Of Single),Nullable(Of Single))`

C#
`public static AggregationView<Nullable<float>,Nullable<float>> LiveMin(` `  View<Nullable<float>> source`

```
)
```

Parameters

source

A view containing the values to determine the minimum value of.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty or contains only nulls, the minimum value is null.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Single>>) Method
The type of the elements of *source*.

A view containing the values to determine the minimum value of.

A transform function to apply to each element.

Computes the minimum value of a view of System.Single values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMin(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Single)) _
) As AggregationView(Of TSource,Single)
```

C#

```
public static AggregationView<TSource,float> LiveMin<TSource>(
   View<TSource> source,
   Expression<Func<TSource,float>> selector
)
```

Parameters

source

A view containing the values to determine the minimum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Single>>>) Method
The type of the elements of *source*.

A view containing the values to determine the minimum value of.

A transform function to apply to each element.

Computes the minimum value of a view of nullable System.Single values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMin(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Single))) _
) As AggregationView(Of TSource,Nullable(Of Single))
```

C#

```
public static AggregationView<TSource,Nullable<float>> LiveMin<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<float>>> selector
)
```

Parameters

source

A view containing the values to determine the minimum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty or contains only nulls, the minimum value is null.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin<TSource,TResult>(View<TSource>,Expression<Func<TSource,TResult>>) Method
The type of the elements of *source*.

The type of the value returned by *selector*.

A view containing the values to determine the minimum value of.

A transform function to apply to each element.

Invokes a transform function on each element of a view of elements of a generic type and computes the minimum resulting value.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMin
    (Of TSource,TResult)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,TResult)) _
) As AggregationView(Of TSource,TResult)
```

C#

```
public static AggregationView<TSource,TResult> LiveMin<TSource,TResult>(
   View<TSource> source,
   Expression<Func<TSource,TResult>> selector
)
```

Parameters

source

A view containing the values to determine the minimum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

TResult

The type of the value returned by *selector*.

Return Value

A view representing the minimum of the values.

Remarks

If type *TResult* implements IComparable, this method uses that implementation to compare values. Otherwise, if type *TResult* implements System.IComparable, that implementation is used to compare values.

It is possible to use standard LINQ query operator **Min** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin<TSource>(View<TSource>) Method
The type of the elements of *source*.

A view containing the values to determine the minimum value of.

Computes the minimum value of a view of elements of a generic type.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMin(Of TSource)( _
   ByVal source As View(Of TSource) _
) As AggregationView(Of TSource,TSource)
```

C#

```
public static AggregationView<TSource,TSource> LiveMin<TSource>(
   View<TSource> source
```

```
)
```

Parameters

source

A view containing the values to determine the minimum value of.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the minimum of the values.

Remarks

If type *TSource* implements IComparable, this method uses that implementation to compare values. Otherwise, if type *TSource* implements System.IComparable, that implementation is used to compare values.

It is possible to use standard LINQ query operator **Min** instead of **LiveMax**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMax** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin(View<Int32>) Method

A view containing the values to determine the minimum value of.

Computes the minimum value of a view of System.Int32 values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMin( _
   ByVal source As View(Of Integer) _
) As AggregationView(Of Integer,Integer)
```

C#

```
public static AggregationView<int,int> LiveMin(
   View<int> source
)
```

Parameters

source

A view containing the values to determine the minimum value of.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty or contains only nulls, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin(View<Nullable<Int32>>) Method
A view containing the values to determine the minimum value of.

Computes the minimum value of a view of nullable System.Int32 values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMin( _
   ByVal source As View(Of Nullable(Of Integer)) _
) As AggregationView(Of Nullable(Of Integer),Nullable(Of Integer))
```

C#

```
public static AggregationView<Nullable<int>,Nullable<int>> LiveMin(
   View<Nullable<int>> source
)
```

Parameters

source

A view containing the values to determine the minimum value of.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty or contains only nulls, the minimum value is null.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate

it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Int32>>) Method
The type of the elements of *source*.

A view containing the values to determine the minimum value of.

A transform function to apply to each element.

Computes the minimum value of a view of System.Int32 values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMin(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Integer)) _
) As AggregationView(Of TSource,Integer)
```

C#

```
public static AggregationView<TSource,int> LiveMin<TSource>(
   View<TSource> source,
   Expression<Func<TSource,int>> selector
)
```

Parameters

source

A view containing the values to determine the minimum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Int32>>>) Method
The type of the elements of *source*.

A view containing the values to determine the minimum value of.

A transform function to apply to each element.

Computes the minimum value of a view of nullable System.Int32 values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMin(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Integer))) _
) As AggregationView(Of TSource,Nullable(Of Integer))
```

C#

```
public static AggregationView<TSource,Nullable<int>> LiveMin<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<int>>> selector
)
```

Parameters

source

A view containing the values to determine the minimum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty or contains only nulls, the minimum value is null.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin(View<Decimal>) Method
A view containing the values to determine the minimum value of.

Computes the minimum value of a view of System.Decimal values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMin( _
   ByVal source As View(Of Decimal) _
) As AggregationView(Of Decimal,Decimal)
```

C#

```
public static AggregationView<decimal,decimal> LiveMin(
   View<decimal> source
)
```

Parameters

source

A view containing the values to determine the minimum value of.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty or contains only nulls, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin(View<Nullable<Decimal>>) Method
A view containing the values to determine the minimum value of.

Computes the minimum value of a view of nullable System.Decimal values.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function LiveMin( _`
`  ByVal source As View(Of Nullable(Of Decimal)) _`
`) As AggregationView(Of Nullable(Of Decimal),Nullable(Of Decimal))`

C#
`public static AggregationView<Nullable<decimal>,Nullable<decimal>> LiveMin(`

```
   View<Nullable<decimal>> source
)
```

Parameters

source

A view containing the values to determine the minimum value of.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty or contains only nulls, the minimum value is null.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Decimal>>) Method
The type of the elements of *source*.

A view containing the values to determine the minimum value of.

A transform function to apply to each element.

Computes the minimum value of a view of System.Decimal values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMin(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Decimal)) _
) As AggregationView(Of TSource,Decimal)
```

C#

```
public static AggregationView<TSource,decimal> LiveMin<TSource>(
   View<TSource> source,
   Expression<Func<TSource,decimal>> selector
)
```

Parameters

source

A view containing the values to determine the minimum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Decimal>>>) Method
The type of the elements of *source*.

A view containing the values to determine the minimum value of.

A transform function to apply to each element.

Computes the minimum value of a view of nullable System.Decimal values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMin(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Decimal))) _
) As AggregationView(Of TSource,Nullable(Of Decimal))
```

C#

```
public static AggregationView<TSource,Nullable<decimal>> LiveMin<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<decimal>>> selector
)
```

Parameters

source

A view containing the values to determine the minimum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty or contains only nulls, the minimum value is null.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin(View<Int64>) Method
A view containing the values to determine the minimum value of.

Computes the minimum value of a view of System.Int64 values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMin( _
   ByVal source As View(Of Long) _
) As AggregationView(Of Long,Long)
```

C#

```
public static AggregationView<long,long> LiveMin(
   View<long> source
)
```

Parameters

source

A view containing the values to determine the minimum value of.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty or contains only nulls, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin(View<Nullable<Int64>>) Method

A view containing the values to determine the minimum value of.

Computes the minimum value of a view of nullable System.Int64 values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMin( _
   ByVal source As View(Of Nullable(Of Long)) _
) As AggregationView(Of Nullable(Of Long),Nullable(Of Long))
```

C#

```
public static AggregationView<Nullable<long>,Nullable<long>> LiveMin(
   View<Nullable<long>> source
)
```

Parameters

source

A view containing the values to determine the minimum value of.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty or contains only nulls, the minimum value is null.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Int64>>) Method
The type of the elements of *source*.

A view containing the values to determine the minimum value of.

A transform function to apply to each element.

Computes the minimum value of a view of System.Int64 values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMin(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Long)) _
) As AggregationView(Of TSource,Long)
```

C#

```
public static AggregationView<TSource,long> LiveMin<TSource>(
   View<TSource> source,
   Expression<Func<TSource,long>> selector
)
```

Parameters

source

A view containing the values to determine the minimum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Int64>>>) Method
The type of the elements of *source*.

A view containing the values to determine the minimum value of.

A transform function to apply to each element.

Computes the minimum value of a view of nullable System.Int64 values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMin(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Long))) _
) As AggregationView(Of TSource,Nullable(Of Long))
```

C#

```
public static AggregationView<TSource,Nullable<long>> LiveMin<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<long>>> selector
)
```

Parameters

source

A view containing the values to determine the minimum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty or contains only nulls, the minimum value is null.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin(View<Double>) Method

A view containing the values to determine the minimum value of.

Computes the minimum value of a view of System.Double values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMin( _
   ByVal source As View(Of Double) _
) As AggregationView(Of Double,Double)
```

C#

```
public static AggregationView<double,double> LiveMin(
   View<double> source
)
```

Parameters

source

A view containing the values to determine the minimum value of.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty or contains only nulls, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin(View<Nullable<Double>>) Method
A view containing the values to determine the minimum value of.

Computes the minimum value of a view of nullable System.Double values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveMin( _
   ByVal source As View(Of Nullable(Of Double)) _
) As AggregationView(Of Nullable(Of Double),Nullable(Of Double))
```

C#

```
public static AggregationView<Nullable<double>,Nullable<double>> LiveMin(
   View<Nullable<double>> source
)
```

Parameters

source

A view containing the values to determine the minimum value of.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty or contains only nulls, the minimum value is null.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveMin<TSource>(View<TSource>,Expression<Func<TSource,Double>>) Method
The type of the elements of *source*.

A view containing the values to determine the minimum value of.

A transform function to apply to each element.

Computes the minimum value of a view of System.Double values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)	
`Public Overloads Shared Function LiveMin(Of TSource)( _` `  ByVal source As View(Of TSource), _`	

```
   ByVal selector As Expression(Of Func(Of TSource,Double)) _
) As AggregationView(Of TSource,Double)
```

C#

```
public static AggregationView<TSource,double> LiveMin<TSource>(
   View<TSource> source,
   Expression<Func<TSource,double>> selector
)
```

Parameters

source

A view containing the values to determine the minimum value of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the minimum of the values.

Remarks

If the *source* is empty, an System.InvalidOperationException is thrown.

It is possible to use standard LINQ query operator **Min** instead of **LiveMin**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Min** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveMin** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveSum Method
Computes the sum of a view of System.Int32 values.

Overload List

Overload	Description
LiveSum(View<Int32>)	Computes the sum of a view of System.Int32 values.
LiveSum(View<Nullable<Int32>>)	Computes the sum of a view of nullable System.Int32 values.
LiveSum<TSource>(View<TSource>,Expression<Func<TSource,Int32>>)	Computes the sum of a view of System.Int32 values that are obtained by invoking a transform function on each element

	of the source view.
LiveSum<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Int32>>>)	Computes the sum of a view of nullable System.Int32 values that are obtained by invoking a transform function on each element of the source view.
LiveSum(View<Decimal>)	Computes the sum of a view of System.Decimal values.
LiveSum(View<Nullable<Decimal>>)	Computes the sum of a view of nullable System.Decimal values.
LiveSum<TSource>(View<TSource>,Expression<Func<TSource,Decimal>>)	Computes the sum of a view of System.Decimal values that are obtained

	by invoking a transform function on each element of the source view.
LiveSum<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Decimal>>>)	Computes the sum of a view of nullable System.Decimal values that are obtained by invoking a transform function on each element of the source view.
LiveSum(View<Int64>)	Computes the sum of a view of System.Int64 values.
LiveSum(View<Nullable<Int64>>)	Computes the sum of a view of nullable System.Int64 values.
LiveSum<TSource>(View<TSource>,Expression<Func<TSource,Int64>>)	Computes the sum of a view

	of System.Int64 values that are obtained by invoking a transform function on each element of the source view.
LiveSum<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Int64>>>)	Computes the sum of a view of nullable System.Int64 values that are obtained by invoking a transform function on each element of the source view.
LiveSum(View<Double>)	Computes the sum of a view of System.Double values.
LiveSum(View<Nullable<Double>>)	Computes the sum of a view of nullable System.Double

	values.
LiveSum<TSource>(View<TSource>,Expression<Func<TSource,Double>>)	Computes the sum of a view of System.Double values that are obtained by invoking a transform function on each element of the source view.
LiveSum<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Double>>>)	Computes the sum of a view of nullable System.Double values that are obtained by invoking a transform function on each element of the source view.
LiveSum(View<Single>)	Computes the sum of a view of System.Single values.

LiveSum(View<Nullable<Single>>)	Computes the sum of a view of nullable System.Single values.
LiveSum<TSource>(View<TSource>,Expression<Func<TSource,Single>>)	Computes the sum of a view of System.Single values that are obtained by invoking a transform function on each element of the source view.
LiveSum<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Single>>>)	Computes the sum of a view of nullable System.Single values that are obtained by invoking a transform function on each element of the source view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members

LiveSum(View<Int32>) Method
A view containing the values to calculate the sum of.

Computes the sum of a view of System.Int32 values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveSum( _
   ByVal source As View(Of Integer) _
) As AggregationView(Of Integer,Integer)
```

C#

```
public static AggregationView<int,int> LiveSum(
   View<int> source
)
```

Parameters

source

A view containing the values to calculate the sum of.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The

difference is that **Sum** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveSum(View<Nullable<Int32>>) Method
A view containing the values to calculate the sum of.

Computes the sum of a view of nullable System.Int32 values.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function LiveSum( _`
`  ByVal source As View(Of Nullable(Of Integer)) _`
`) As AggregationView(Of Nullable(Of Integer),Nullable(Of Integer))`

C#

`public static AggregationView<Nullable<int>,Nullable<int>> LiveSum(`
`  View<Nullable<int>> source`
`)`

Parameters

source

A view containing the values to calculate the sum of.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveSum<TSource>(View<TSource>,Expression<Func<TSource,Int32>>) Method
The type of the elements of *source*.

A view containing the values to calculate the sum of.

A transform function to apply to each element.

Computes the sum of a view of System.Int32 values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveSum(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Integer)) _
) As AggregationView(Of TSource,Integer)
```

C#

```
public static AggregationView<TSource,int> LiveSum<TSource>(
   View<TSource> source,
   Expression<Func<TSource,int>> selector
)
```

Parameters

source

A view containing the values to calculate the sum of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveSum<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Int32>>>) Method
The type of the elements of *source*.

A view containing the values to calculate the sum of.

A transform function to apply to each element.

Computes the sum of a view of nullable System.Int32 values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveSum(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Integer))) _
) As AggregationView(Of TSource,Nullable(Of Integer))
```

C#

```
public static AggregationView<TSource,Nullable<int>> LiveSum<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<int>>> selector
)
```

Parameters

source

A view containing the values to calculate the sum of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveSum(View<Decimal>) Method
A view containing the values to calculate the sum of.

Computes the sum of a view of System.Decimal values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveSum( _
   ByVal source As View(Of Decimal) _
) As AggregationView(Of Decimal,Decimal)
```

C#

```
public static AggregationView<decimal,decimal> LiveSum(
   View<decimal> source
)
```

Parameters

source

A view containing the values to calculate the sum of.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveSum(View<Nullable<Decimal>>) Method
A view containing the values to calculate the sum of.

Computes the sum of a view of nullable System.Decimal values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveSum( _
   ByVal source As View(Of Nullable(Of Decimal)) _
) As AggregationView(Of Nullable(Of Decimal),Nullable(Of Decimal))
```

C#

```
public static AggregationView<Nullable<decimal>,Nullable<decimal>> LiveSum(
   View<Nullable<decimal>> source
)
```

Parameters

source

A view containing the values to calculate the sum of.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

The type of the elements of *source*.

A view containing the values to calculate the sum of.

A transform function to apply to each element.

Computes the sum of a view of System.Decimal values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveSum(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Decimal)) _
) As AggregationView(Of TSource,Decimal)
```

C#

```
public static AggregationView<TSource,decimal> LiveSum<TSource>(
   View<TSource> source,
   Expression<Func<TSource,decimal>> selector
)
```

Parameters

source

A view containing the values to calculate the sum of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveSum<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Decimal>>>) Method
The type of the elements of *source*.

A view containing the values to calculate the sum of.

A transform function to apply to each element.

Computes the sum of a view of nullable System.Decimal values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveSum(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Decimal))) _
) As AggregationView(Of TSource,Nullable(Of Decimal))
```

C#

```
public static AggregationView<TSource,Nullable<decimal>> LiveSum<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<decimal>>> selector
)
```

Parameters

source

A view containing the values to calculate the sum of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveSum(View<Int64>) Method

A view containing the values to calculate the sum of.

Computes the sum of a view of System.Int64 values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveSum( _
   ByVal source As View(Of Long) _
) As AggregationView(Of Long,Long)
```

C#

```
public static AggregationView<long,long> LiveSum(
   View<long> source
)
```

Parameters

source

A view containing the values to calculate the sum of.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveSum(View<Nullable<Int64>>) Method
A view containing the values to calculate the sum of.

Computes the sum of a view of nullable System.Int64 values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveSum( _
   ByVal source As View(Of Nullable(Of Long)) _
) As AggregationView(Of Nullable(Of Long),Nullable(Of Long))
```

C#

```
public static AggregationView<Nullable<long>,Nullable<long>> LiveSum(
   View<Nullable<long>> source
)
```

Parameters

source

A view containing the values to calculate the sum of.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveSum<TSource>(View<TSource>,Expression<Func<TSource,Int64>>) Method
The type of the elements of *source*.

A view containing the values to calculate the sum of.

A transform function to apply to each element.

Computes the sum of a view of System.Int64 values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveSum(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Long)) _
) As AggregationView(Of TSource,Long)
```

C#

```
public static AggregationView<TSource,long> LiveSum<TSource>(
   View<TSource> source,
   Expression<Func<TSource,long>> selector
```

```
)
```

Parameters

source

A view containing the values to calculate the sum of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

The type of the elements of *source*.

A view containing the values to calculate the sum of.

A transform function to apply to each element.

Computes the sum of a view of nullable System.Int64 values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveSum(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Long))) _
) As AggregationView(Of TSource,Nullable(Of Long))
```

C#

```
public static AggregationView<TSource,Nullable<long>> LiveSum<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<long>>> selector
)
```

Parameters

source

A view containing the values to calculate the sum of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveSum(View<Double>) Method
A view containing the values to calculate the sum of.

Computes the sum of a view of System.Double values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveSum( _
   ByVal source As View(Of Double) _
) As AggregationView(Of Double,Double)
```

C#

```
public static AggregationView<double,double> LiveSum(
   View<double> source
)
```

Parameters

source

A view containing the values to calculate the sum of.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveSum(View<Nullable<Double>>) Method
A view containing the values to calculate the sum of.

Computes the sum of a view of nullable System.Double values.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function LiveSum( _`
`  ByVal source As View(Of Nullable(Of Double)) _`
`) As AggregationView(Of Nullable(Of Double),Nullable(Of Double))`

C#

```
public static AggregationView<Nullable<double>,Nullable<double>> LiveSum(
   View<Nullable<double>> source
)
```

Parameters

source

A view containing the values to calculate the sum of.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveSum<TSource>(View<TSource>,Expression<Func<TSource,Double>>) Method
The type of the elements of *source*.

A view containing the values to calculate the sum of.

A transform function to apply to each element.

Computes the sum of a view of System.Double values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveSum(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Double)) _
) As AggregationView(Of TSource,Double)
```

C#

```
public static AggregationView<TSource,double> LiveSum<TSource>(
   View<TSource> source,
   Expression<Func<TSource,double>> selector
)
```

Parameters

source

A view containing the values to calculate the sum of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the

entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveSum<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Double>>>) Method
The type of the elements of *source*.

A view containing the values to calculate the sum of.

A transform function to apply to each element.

Computes the sum of a view of nullable System.Double values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveSum(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Double))) _
) As AggregationView(Of TSource,Nullable(Of Double))
```

C#

```
public static AggregationView<TSource,Nullable<double>> LiveSum<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<double>>> selector
)
```

Parameters

source

A view containing the values to calculate the sum of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveSum(View<Single>) Method
A view containing the values to calculate the sum of.

Computes the sum of a view of System.Single values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveSum( _
   ByVal source As View(Of Single) _
) As AggregationView(Of Single,Single)
```

C#

```
public static AggregationView<float,float> LiveSum(
   View<float> source
)
```

Parameters

source

A view containing the values to calculate the sum of.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveSum(View<Nullable<Single>>) Method
A view containing the values to calculate the sum of.

Computes the sum of a view of nullable System.Single values.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveSum( _
   ByVal source As View(Of Nullable(Of Single)) _
) As AggregationView(Of Nullable(Of Single),Nullable(Of Single))
```

C#

```
public static AggregationView<Nullable<float>,Nullable<float>> LiveSum(
   View<Nullable<float>> source
)
```

Parameters

source

A view containing the values to calculate the sum of.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveSum<TSource>(View<TSource>,Expression<Func<TSource,Single>>) Method
The type of the elements of *source*.

A view containing the values to calculate the sum of.

A transform function to apply to each element.

Computes the sum of a view of System.Single values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function LiveSum(Of TSource)( _`
`  ByVal source As View(Of TSource), _`
`  ByVal selector As Expression(Of Func(Of TSource,Single)) _`
`) As AggregationView(Of TSource,Single)`

C#

`public static AggregationView<TSource,float> LiveSum<TSource>(`
`  View<TSource> source,`
`  Expression<Func<TSource,float>> selector`
`)`

Parameters

source

A view containing the values to calculate the sum of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

LiveSum<TSource>(View<TSource>,Expression<Func<TSource,Nullable<Single>>>) Method
The type of the elements of *source*.

A view containing the values to calculate the sum of.

A transform function to apply to each element.

Computes the sum of a view of nullable System.Single values that are obtained by invoking a transform function on each element of the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function LiveSum(Of TSource)( _
   ByVal source As View(Of TSource), _
   ByVal selector As Expression(Of Func(Of TSource,Nullable(Of Single))) _
) As AggregationView(Of TSource,Nullable(Of Single))
```

C#

```
public static AggregationView<TSource,Nullable<float>> LiveSum<TSource>(
   View<TSource> source,
   Expression<Func<TSource,Nullable<float>>> selector
)
```

Parameters

source

A view containing the values to calculate the sum of.

selector

A transform function to apply to each element.

Type Parameters

TSource

The type of the elements of *source*.

Return Value

A view representing the sum of the values.

Remarks

If the *source* is empty or contains only nulls, the sum value is zero.

It is possible to use standard LINQ query operator **Sum** instead of **LiveSum**. Both are "live" in the sense that they are recomputed automatically when any change occurs in the source. The difference is that **Sum** will every time loop through the entire source collection and aggregate it from scratch, whereas **LiveSum** will use a more performant algorithm, will maintain its value incrementally, processing only those source items that actually changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

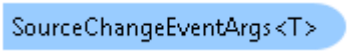
Reference

LiveViewExtensions Class
LiveViewExtensions Members
Overload List

SourceChangeEventArgs<T>
Type of the changed object.

Provides data for the IObservableSource<T>.Changed event.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class SourceChangeEventArgs(Of T)` `  Inherits System.EventArgs`
C#
`public class SourceChangeEventArgs<T> : System.EventArgs`

Type Parameters

T

Type of the changed object.

Inheritance Hierarchy

System.Object
  System.EventArgs
    C1.LiveLinq.SourceChangeEventArgs<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SourceChangeEventArgs<T> Members
C1.LiveLinq Namespace

Overview
Type of the changed object.

Provides data for the IObservableSource<T>.Changed event.

Object Model

Syntax

Visual Basic (Declaration)
`Public Class SourceChangeEventArgs(Of T)` `  Inherits System.EventArgs`
C#
`public class SourceChangeEventArgs<T> : System.EventArgs`

Type Parameters

T

Type of the changed object.

Inheritance Hierarchy

System.Object
  System.EventArgs
    C1.LiveLinq.SourceChangeEventArgs<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SourceChangeEventArgs<T> Members
C1.LiveLinq Namespace

Members
Properties Methods

The following tables list the members exposed by SourceChangeEventArgs<T>.

Public Constructors

	Name	Description
	SourceChangeEventArgs<T> Constructor	Initializes a new instance of the SourceChangeEventArgs<T> class.

Top

Public Properties

	Name	Description
	ChangeType	Gets the type of change.
	Item	Gets the object that is being changed.
	Ordinal	Gets the ordinal position of the collection item that is being changed.

Top

Public Methods

	Name	Description
	Equals	Overloaded. Determines whether the specified object is equal to the current object.
	GetHashCode	Return a hash code for this instance.
	ToString	Returns a string that represents this instance of SourceChangeEventArgs<T>.

Top

See Also

Reference

SourceChangeEventArgs<T> Class
C1.LiveLinq Namespace

SourceChangeEventArgs<T> Constructor
Changed object.

Type of change.

Ordinal position of the changed item.

Initializes a new instance of the SourceChangeEventArgs<T> class.

Syntax

Visual Basic (Declaration)

```
Public Function New( _
   ByVal item As T, _
   ByVal changeType As SourceChangeType, _
   ByVal ordinal As Integer _
)
```

C#

```
public SourceChangeEventArgs<T>(
   T item,
   SourceChangeType changeType,
   int ordinal
)
```

Parameters

item

Changed object.

changeType

Type of change.

ordinal

Ordinal position of the changed item.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SourceChangeEventArgs<T> Class
SourceChangeEventArgs<T> Members

Methods

For a list of all members of this type, see SourceChangeEventArgs<T> members.

Public Methods

	Name	Description
	Equals	Overloaded. Determines whether the specified object is equal to the current object.

	GetHashCode	Return a hash code for this instance.
	ToString	Returns a string that represents this instance of SourceChangeEventArgs<T>.

Top

See Also

Reference

SourceChangeEventArgs<T> Class
C1.LiveLinq Namespace

Equals Method
Determines whether the specified object is equal to the current object.

Overload List

Overload	Description
Equals(Object)	Determines whether the specified object is equal to the current object.
Equals(SourceChangeEventArgs<T>)	Determines whether the specified object is equal to the current object.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SourceChangeEventArgs<T> Class
SourceChangeEventArgs<T> Members

Equals(Object) Method
The object to compare with the current object.

Determines whether the specified object is equal to the current object.

Syntax

Visual Basic (Declaration)

```
Public Overloads Overrides Function Equals( _
   ByVal obj As Object _
) As Boolean
```

C#

```
public override bool Equals(
   object obj
)
```

Parameters

obj

The object to compare with the current object.

Return Value

true if the specified object is equal to the current one; otherwise, false.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SourceChangeEventArgs<T> Class
SourceChangeEventArgs<T> Members
Overload List

Equals(SourceChangeEventArgs<T>) Method
The object to compare with the current object.

Determines whether the specified object is equal to the current object.

Syntax

Visual Basic (Declaration)	

`Public Overloads Function Equals( _`
`  ByVal args As SourceChangeEventArgs(Of T) _`
`) As Boolean`	

C#	

`public bool Equals(`
`  SourceChangeEventArgs<T> args`
`)`	

Parameters

args

The object to compare with the current object.

Return Value

true if the specified object is equal to the current one; otherwise, false.

Remarks

Two SourceChangeEventArgs<T> are considered equal if the values of their Item and ChangeType properties are equal.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SourceChangeEventArgs<T> Class
SourceChangeEventArgs<T> Members
Overload List

GetHashCode Method
Return a hash code for this instance.

Syntax

Visual Basic (Declaration)
`Public Overrides Function GetHashCode() As Integer`
C#
`public override int GetHashCode()`

Return Value

A 32-bit signed integer hash code.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SourceChangeEventArgs<T> Class
SourceChangeEventArgs<T> Members

ToString Method
Returns a string that represents this instance of SourceChangeEventArgs<T>.

Syntax

Visual Basic (Declaration)
`Public Overrides Function ToString() As String`
C#
`public override string ToString()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SourceChangeEventArgs<T> Class
SourceChangeEventArgs<T> Members

Properties
For a list of all members of this type, see SourceChangeEventArgs<T> members.

Public Properties

	Name	Description
	ChangeType	Gets the type of change.
	Item	Gets the object that is being changed.
	Ordinal	Gets the ordinal position of the collection item that is being changed.

Top

See Also

Reference

SourceChangeEventArgs<T> Class
C1.LiveLinq Namespace

ChangeType Property
Gets the type of change.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property ChangeType As SourceChangeType`
C#
`public SourceChangeType ChangeType {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SourceChangeEventArgs<T> Class
SourceChangeEventArgs<T> Members

Item Property
Gets the object that is being changed.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Item As T`
C#
`public T Item {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SourceChangeEventArgs<T> Class
SourceChangeEventArgs<T> Members

Ordinal Property
Gets the ordinal position of the collection item that is being changed.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Property Ordinal As Integer
```

C#

```
public int Ordinal {get;}
```

Remarks

This property can return -1 (ordinal unknown) if the collection cannot provide this information (if IObservableSource<T>.SupportsItemOrdinals returns **false**).

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

SourceChangeEventArgs<T> Class
SourceChangeEventArgs<T> Members

Enumerations

Order
Indicates if a certain order is required in the result collection of an operation.

Syntax

Visual Basic (Declaration)

```
Public Enum Order
   Inherits System.Enum
```

C#

```
public enum Order : System.Enum
```

Members

Member	Description

Ascending	The resulting collection must be ordered in ascending key order.
Descending	The resulting collection must be ordered in descending key order.
Unordered	No particular order is required in the resulting collection.

Inheritance Hierarchy

System.Object
  System.ValueType
    System.Enum
      C1.LiveLinq.Order

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1.LiveLinq Namespace

SourceChangeType
Describes a change occurring in a collection.

Syntax

Visual Basic (Declaration)

```
Public Enum SourceChangeType
   Inherits System.Enum
```

C#

```
public enum SourceChangeType : System.Enum
```

Members

Member	Description
Add	An item is added to the collection.
Modify	At least one of the properties of an item has changed its value.
Remove	An item is removed from the collection.
Reset	Multiple items have changed in the collection. Indexes must be rebuilt.

Inheritance Hierarchy

System.Object
  System.ValueType
    System.Enum
      C1.LiveLinq.SourceChangeType

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1.LiveLinq Namespace

TransactionState
Enumeration of the possible states an ITransaction can be in.

Syntax

Visual Basic (Declaration)
`Public Enum TransactionState` `  Inherits System.Enum`
C#

```
public enum TransactionState : System.Enum
```

Members

Member	Description
Committed	A transaction is committed.
Committing	A transaction is in the process of being committed.
Open	A transaction is open.
RolledBack	A transaction is rolled back.
RollingBack	A transaction is in the process of being rolled back.

Inheritance Hierarchy

System.Object
  System.ValueType
    System.Enum
      C1.LiveLinq.TransactionState

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1.LiveLinq Namespace

Interfaces

IObservableSource<T>
The type of the elements in the collection.

Provides methods and events that are required for LiveLinq functionality, live views.

Object Model

IObservableSource<T>

Syntax

Visual Basic (Declaration)
`Public Interface IObservableSource(Of T)`
C#
`public interface IObservableSource<T>`

Type Parameters

T

The type of the elements in the collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IObservableSource<T> Members
C1.LiveLinq Namespace

Overview
The type of the elements in the collection.

Provides methods and events that are required for LiveLinq functionality, live views.

Object Model

IObservableSource<T>

Syntax

Visual Basic (Declaration)
`Public Interface IObservableSource(Of T)`
C#
`public interface IObservableSource<T>`

Type Parameters

T

The type of the elements in the collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IObservableSource<T> Members
C1.LiveLinq Namespace

Members
Properties Methods Events

The following tables list the members exposed by IObservableSource<T>.

Public Properties

	Name	Description
	CreateNew	This delegate is used to create new items. If it is null, a public parameterless constructor of type **T** is used.
	IsDeletedStateAvailable	Gets a value indicating whether an item of this collection can still return correct property values after it has been deleted from the collection.

	SupportsItemOrdinals	Gets a value that indicates whether this collection is capable of providing the ordinal position of the changed item when it notifies of an item change.

Top

Public Methods

	Name	Description
	EnableItemOrdinals	After this method is called, the collection is required to provide SourceChangeEventArgs<T>.Ordinal in event data.

Top

Public Events

	Name	Description
	Changed	Occurs after an item of the collection or the entire collection has changed.

Top

See Also

Reference

IObservableSource<T> Interface
C1.LiveLinq Namespace

Methods
For a list of all members of this type, see IObservableSource<T> members.

Public Methods

	Name	Description
	EnableItemOrdinals	After this method is called, the collection is required to provide SourceChangeEventArgs<T>.Ordinal in event data.

Top

See Also

Reference

IObservableSource<T> Interface
C1.LiveLinq Namespace

EnableItemOrdinals Method
After this method is called, the collection is required to provide SourceChangeEventArgs<T>.Ordinal in event data.

Syntax

Visual Basic (Declaration)
`Sub EnableItemOrdinals()`
C#
`void EnableItemOrdinals()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IObservableSource<T> Interface
IObservableSource<T> Members

Properties
For a list of all members of this type, see IObservableSource<T> members.

Public Properties

	Name	Description
	CreateNew	This delegate is used to create new items. If it is null, a public parameterless constructor of type **T** is used.

	IsDeletedStateAvailable	Gets a value indicating whether an item of this collection can still return correct property values after it has been deleted from the collection.
	SupportsItemOrdinals	Gets a value that indicates whether this collection is capable of providing the ordinal position of the changed item when it notifies of an item change.

Top

See Also

Reference

IObservableSource<T> Interface
C1.LiveLinq Namespace

CreateNew Property
This delegate is used to create new items. If it is null, a public parameterless constructor of type **T** is used.

Syntax

Visual Basic (Declaration)
`ReadOnly Property CreateNew As Func(Of T)`
C#
`Func<T> CreateNew {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IObservableSource<T> Interface
IObservableSource<T> Members

IsDeletedStateAvailable Property

Gets a value indicating whether an item of this collection can still return correct property values after it has been deleted from the collection.

Syntax

Visual Basic (Declaration)
`ReadOnly Property IsDeletedStateAvailable As Boolean`
C#
`bool IsDeletedStateAvailable {get;}`

Property Value

true if deleted items can still be used to get property values.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IObservableSource<T> Interface
IObservableSource<T> Members

SupportsItemOrdinals Property

Gets a value that indicates whether this collection is capable of providing the ordinal position of the changed item when it notifies of an item change.

Syntax

Visual Basic (Declaration)
`ReadOnly Property SupportsItemOrdinals As Boolean`
C#
`bool SupportsItemOrdinals {get;}`

Property Value

true if the collection can provide item ordinal positions.

Remarks

If this property returns **true**, LiveLinq can call the EnableItemOrdinals method to require providing ordinals.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

IObservableSource<T> Interface
IObservableSource<T> Members

Events
>

Name	Description
Changed	Occurs after an item of the collection or the entire collection has changed.

Top

See Also

Reference

IObservableSource<T> Interface
C1.LiveLinq Namespace

Changed Event
Occurs after an item of the collection or the entire collection has changed.

Syntax

Visual Basic (Declaration)	

```
Event Changed As EventHandler(Of SourceChangeEventArgs(Of T))
```

C#

```
event EventHandler<SourceChangeEventArgs<T>> Changed
```

Event Data

The event handler receives an argument of type SourceChangeEventArgs<T> containing data related to this event. The following **SourceChangeEventArgs<T>** properties provide information specific to this event.

Property	Description
ChangeType	Gets the type of change.
Item	Gets the object that is being changed.
Ordinal	Gets the ordinal position of the collection item that is being changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

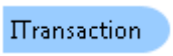
See Also

Reference

IObservableSource<T> Interface
IObservableSource<T> Members

ITransaction
Represents a transaction with an explicit scope.

Object Model



Syntax

Visual Basic (Declaration)	
`Public Interface ITransaction`	
C#	
`public interface ITransaction`	

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

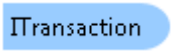
See Also

Reference

ITransaction Members
C1.LiveLinq Namespace

Overview
Represents a transaction with an explicit scope.

Object Model



Syntax

Visual Basic (Declaration)	
`Public Interface ITransaction`	
C#	
`public interface ITransaction`	

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ITransaction Members
C1.LiveLinq Namespace

Members
Properties Methods

The following tables list the members exposed by ITransaction.

Public Properties

	Name	Description
	HasChanges	Gets a value indicating whether any changes were made in the scope of this transaction.
	State	Gets the current state of the transaction.

Top

Public Methods

	Name	Description
	Commit	Commits changes that were made while this transaction's scope was open.
	Rollback	Rolls back changes that were made while this transaction's scope was open.
	Scope	Opens a transaction scope.

Top

See Also

Reference

ITransaction Interface
C1.LiveLinq Namespace

Methods

For a list of all members of this type, see ITransaction members.

Public Methods

	Name	Description
	Commit	Commits changes that were made while this transaction's scope was open.
	Rollback	Rolls back changes that were made while this transaction's scope was open.
	Scope	Opens a transaction scope.

Top

See Also

Reference

ITransaction Interface
C1.LiveLinq Namespace

Commit Method

Commits changes that were made while this transaction's scope was open.

Syntax

Visual Basic (Declaration)
`Sub Commit()`
C#
`void Commit()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ITransaction Interface
ITransaction Members

Rollback Method
Rolls back changes that were made while this transaction's scope was open.

Syntax

Visual Basic (Declaration)
`Sub Rollback()`
C#
`void Rollback()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ITransaction Interface
ITransaction Members

Scope Method
Opens a transaction scope.

Syntax

Visual Basic (Declaration)
`Function Scope() As IDisposable`
C#
`IDisposable Scope()`

Return Value

An instance of System.IDisposable that will close the scope when its System.IDisposable.Dispose method is called.

Remarks

The transaction tracks changes only when they are made inside an open scope.

Calling System.IDisposable.Dispose on the return value closes the scope.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ITransaction Interface
ITransaction Members

Properties
For a list of all members of this type, see ITransaction members.

Public Properties

	Name	Description
	HasChanges	Gets a value indicating whether any changes were made in the scope of this transaction.
	State	Gets the current state of the transaction.

Top

See Also

Reference

ITransaction Interface
C1.LiveLinq Namespace

HasChanges Property

Gets a value indicating whether any changes were made in the scope of this transaction.

Syntax

Visual Basic (Declaration)
`ReadOnly Property HasChanges As Boolean`
C#
`bool HasChanges {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ITransaction Interface
ITransaction Members

State Property

Gets the current state of the transaction.

Syntax

Visual Basic (Declaration)
`ReadOnly Property State As TransactionState`
C#
`TransactionState State {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ITransaction Interface
ITransaction Members

C1.LiveLinq.LiveViews Namespace

Overview

Classes

	Class	Description
	AggregationView<TSource,TResult>	Represents a view having a single element calculated by aggregating a source view.
	GroupView<TKey,TElement>	A group in a grouping view.
	OrderedView<T>	Represents a sorted view.
	PropertyIsNotVirtualException	Represents an exception that indicates that a property used in a result selector of a live view is not virtual.
	View	Base class for the View<T> class. Contains members that don't depend on the element type *T*.
	View<T>	Represents a *live view*: a LINQ query result that supports two-way data binding and is kept up-to-date with base data.
	ViewRow	Represents a view element (item) for the purposes of dynamic, programmatic access to its properties and data binding.
	ViewRowAddingEventArgs	Provides data for the ViewRowCollection.ViewRowAdding event.

	ViewRowCollection	Represents a collection of ViewRow objects used for programmatic access to view elements (items) and for data binding.
	ViewRowPropertyInfo	Allows to control certain behavior of a property of the element type of a View.

Enumerations

	Enumeration	Description
	ViewMaintenanceMode	Specifies how a view is synchronized with changes in its base data.
	ViewOrder	Specifies whether and how a view must preserve item order if it exists in the source.
	ViewRowState	The state of a view row with regard to edit, add and delete operations if they are performed directly on the view.

See Also

Reference

C1.Silverlight.LiveLinq Assembly

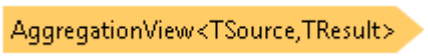
Classes

AggregationView<TSource,TResult>
The type of the elements of the source view.

The type of the single element of the aggregation view.

Represents a view having a single element calculated by aggregating a source view.

Object Model



Syntax

Visual Basic (Declaration)	

`Public Class AggregationView`
`  (Of TSource,TResult)`
`  Inherits View(Of TResult)`
`  Implements C1.LiveLinq.IObservableSource(Of TResult)`	

C#	
`public class AggregationView<TSource,TResult> : View<TResult>, C1.LiveLinq.IObservableSource<TResult>`	

Type Parameters

TSource

The type of the elements of the source view.

TResult

The type of the single element of the aggregation view.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.View
    C1.LiveLinq.LiveViews.View<T>
      C1.LiveLinq.LiveViews.AggregationView<TSource,TResult>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

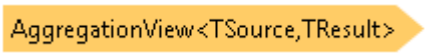
AggregationView<TSource,TResult> Members
C1.LiveLinq.LiveViews Namespace

Overview
The type of the elements of the source view.

The type of the single element of the aggregation view.

Represents a view having a single element calculated by aggregating a source view.

Object Model



Syntax

Visual Basic (Declaration)

`Public Class AggregationView`
`  (Of TSource,TResult)`
`  Inherits View(Of TResult)`
`  Implements C1.LiveLinq.IObservableSource(Of TResult)`

C#
`public class AggregationView<TSource,TResult> : View<TResult>, C1.LiveLinq.IObservableSource<TResult>`

Type Parameters

TSource

The type of the elements of the source view.

TResult

The type of the single element of the aggregation view.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.View
    C1.LiveLinq.LiveViews.View<T>
      C1.LiveLinq.LiveViews.AggregationView<TSource,TResult>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

AggregationView<TSource,TResult> Members
C1.LiveLinq.LiveViews Namespace

Members
Properties Methods Events

The following tables list the members exposed by AggregationView<TSource,TResult>.

Public Properties

	Name	Description
	Count	Gets the total number of elements in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	CurrentItem	Gets the current item in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	DeferredMaintenance	Gets the effective value of MaintenanceMode. (Inherited from C1.LiveLinq.LiveViews.View)
	IsReadOnly	Gets a value indicating whether this view is read-only, not updatable. (Inherited from C1.LiveLinq.LiveViews.View)
	Item	Gets the view item (element) at the specified ordinal position. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)
	MaintenanceMode	Gets or sets a value controlling how the view is synchronized with changes in its base data. (Inherited from C1.LiveLinq.LiveViews.View)
	MoveToFirstOnReset	Gets or sets a value indicating that the first item must be made current after initial loading or reset (on any C1.LiveLinq.SourceChangeType.Reset notification) if current item was not set by other means. The default is True. (Inherited from

		C1.LiveLinq.LiveViews.View)
	Order	Gets a value indicating whether and how this view preserves item order if it exists in its base data source. (Inherited from C1.LiveLinq.LiveViews.View)
	Transaction	Gets an instance of C1.LiveLinq.ITransaction associated with the view. If a view has a transaction associated with it, that transaction's scope is opened automatically every time the view is updated, so the programmer does not need to do it manually in code. (Inherited from C1.LiveLinq.LiveViews.View)
	Value	Gets the value of the single element of the aggregation view.

Top

Public Methods

	Name	Description
	AsCollectionViewFactory	Returns an instance of System.ComponentModel.ICollectionViewFactory that can be used as a source of a System.Windows.Data.CollectionViewSource. (Inherited from C1.LiveLinq.LiveViews.View)
	AsDynamic	Used for views with anonymous type constructor as the result selector, converts the View to a View<dynamic> so it can be used for data binding and programmatic access. (Inherited from C1.LiveLinq.LiveViews.View)
	AttachAggregationView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)
	AttachView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)

	Concat	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)
	Contains	Determines whether the view contains a specified item. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)
	DeferMaintenance	Enters a defer cycle that you can use to make bulk changes to the view sources and delay automatic view maintenance. (Inherited from C1.LiveLinq.LiveViews.View)
	GetEnumerator	Returns an enumerator that iterates through the view items. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)
	GroupBy	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)
	GroupJoin	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)
	IndexOf	Searches for the specified object among the elements of the view and returns the zero-based ordinal position of its first occurrence. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)
	Join	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)
	Maintain	Brings the view up to date with its source data. (Inherited from C1.LiveLinq.LiveViews.View)
	OrderBy<TKey>	Sorts the elements of a view in ascending order. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)
	OrderByDescending<TKey>	Sorts the elements of a view in descending order. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)

	PurgeEmptyGroups	Remove empty groups from a grouping view. (Inherited from C1.LiveLinq.LiveViews.View)
	Rebuild	Re-populates the view by re-executing the view's query. (Inherited from C1.LiveLinq.LiveViews.View)
	Select<TResult>	Projects each element of a view into a new form. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)
	SelectMany	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)
	SetTransaction	Sets the value of the Transaction property. (Inherited from C1.LiveLinq.LiveViews.View)
	ToString	Returns a string representing this view. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)
	Union	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)
	Where	Filters the source view based on a predicate. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)

Top

Public Events

	Name	Description
	Changed	Occurs after an item of the view or the entire view has changed. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)

Top

See Also

Reference

AggregationView<TSource,TResult> Class
C1.LiveLinq.LiveViews Namespace

Properties
For a list of all members of this type, see AggregationView<TSource,TResult> members.

Public Properties

	Name	Description
	Count	Gets the total number of elements in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	CurrentItem	Gets the current item in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	DeferredMaintenance	Gets the effective value of MaintenanceMode. (Inherited from C1.LiveLinq.LiveViews.View)
	IsReadOnly	Gets a value indicating whether this view is read-only, not updatable. (Inherited from C1.LiveLinq.LiveViews.View)
	Item	Gets the view item (element) at the specified ordinal position. (Inherited from C1.LiveLinq.LiveViews.View<TResult>)
	MaintenanceMode	Gets or sets a value controlling how the view is synchronized with changes in its base data. (Inherited from C1.LiveLinq.LiveViews.View)
	MoveToFirstOnReset	Gets or sets a value indicating that the first item must be made current after initial loading or reset (on any C1.LiveLinq.SourceChangeType.Reset notification) if current item was not set by other means. The default is True. (Inherited from C1.LiveLinq.LiveViews.View)
	Order	Gets a value indicating whether and how this view preserves item order

		if it exists in its base data source. (Inherited from C1.LiveLinq.LiveViews.View)
	Transaction	Gets an instance of C1.LiveLinq.ITransaction associated with the view. If a view has a transaction associated with it, that transaction's scope is opened automatically every time the view is updated, so the programmer does not need to do it manually in code. (Inherited from C1.LiveLinq.LiveViews.View)
	Value	Gets the value of the single element of the aggregation view.

Top

See Also

Reference

AggregationView<TSource,TResult> Class
C1.LiveLinq.LiveViews Namespace

Value Property
Gets the value of the single element of the aggregation view.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Property Value As TResult
```

C#

```
public TResult Value {get;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

AggregationView<TSource,TResult> Class
AggregationView<TSource,TResult> Members

GroupView<TKey,TElement>
The type of the key used for grouping.

The type of the elements in the group view.

A group in a grouping view.

Object Model

Syntax

Visual Basic (Declaration)

`Public NotInheritable Class GroupView`
`  (Of TKey,TElement)`
`  Inherits View(Of TElement)`
`  Implements C1.LiveLinq.IObservableSource(Of TElement)`

C#
`public sealed class GroupView<TKey,TElement> : View<TElement>, C1.LiveLinq.IObservableSource<TElement>`

Type Parameters

TKey

The type of the key used for grouping.

TElement

The type of the elements in the group view.

Remarks

A grouping view is a result of a **GroupBy** operation on a live view. It is a collection of groups. Each group contains elements with the same key. That collection is a live view, and every group in itself is a live view, an object of type **GroupView<TKey, TElement>**.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.View
    C1.LiveLinq.LiveViews.View<T>
      C1.LiveLinq.LiveViews.GroupView<TKey,TElement>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

GroupView<TKey,TElement> Members
C1.LiveLinq.LiveViews Namespace

Overview
The type of the key used for grouping.

The type of the elements in the group view.

A group in a grouping view.

Object Model

Syntax

Visual Basic (Declaration)

`Public NotInheritable Class GroupView`
`  (Of TKey,TElement)`
`  Inherits View(Of TElement)`
`  Implements C1.LiveLinq.IObservableSource(Of TElement)`

C#
`public sealed class GroupView<TKey,TElement> : View<TElement>,`

```
C1.LiveLinq.IObservableSource<TElement>
```

Type Parameters

TKey

The type of the key used for grouping.

TElement

The type of the elements in the group view.

Remarks

A grouping view is a result of a **GroupBy** operation on a live view. It is a collection of groups. Each group contains elements with the same key. That collection is a live view, and every group in itself is a live view, an object of type **GroupView<TKey, TElement>**.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.View
    C1.LiveLinq.LiveViews.View<T>
      C1.LiveLinq.LiveViews.GroupView<TKey,TElement>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

GroupView<TKey,TElement> Members
C1.LiveLinq.LiveViews Namespace

Members
Properties Methods Events

The following tables list the members exposed by GroupView<TKey,TElement>.

Public Properties

	Name	Description
	Count	Gets the total number of elements in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	CurrentItem	Gets the current item in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	DeferredMaintenance	Overridden. This property overrides DeferredMaintenance.
	IsReadOnly	Gets a value indicating whether this view is read-only, not updatable. (Inherited from C1.LiveLinq.LiveViews.View)
	Item	Gets the view item (element) at the specified ordinal position. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
	Key	Gets the key value of the group.
	MaintenanceMode	Overridden. This property overrides MaintenanceMode.
	MoveToFirstOnReset	Gets or sets a value indicating that the first item must be made current after initial loading or reset (on any C1.LiveLinq.SourceChangeType.Reset notification) if current item was not set by other means. The default is True. (Inherited from C1.LiveLinq.LiveViews.View)
	Order	Gets a value indicating whether and how this view preserves item order if it exists in its base data source. (Inherited from C1.LiveLinq.LiveViews.View)
	Parent	Gets the grouping view (the result of a **GroupBy** operation) to which this group belongs.
	Transaction	Gets an instance of C1.LiveLinq.ITransaction associated with the view. If a view has a transaction associated with it, that transaction's scope is

		opened automatically every time the view is updated, so the programmer does not need to do it manually in code. (Inherited from C1.LiveLinq.LiveViews.View)

Top

Public Methods

	Name	Description
	AsCollectionViewFactory	Returns an instance of System.ComponentModel.ICollectionViewFactory that can be used as a source of a System.Windows.Data.CollectionViewSource. (Inherited from C1.LiveLinq.LiveViews.View)
	AsDynamic	Used for views with anonymous type constructor as the result selector, converts the View to a View<dynamic> so it can be used for data binding and programmatic access. (Inherited from C1.LiveLinq.LiveViews.View)
	AttachAggregationView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
	AttachView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
	Concat	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
	Contains	Determines whether the view contains a specified item. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
	DeferMaintenance	Enters a defer cycle that you can use to make bulk changes to the view sources and delay automatic view maintenance. (Inherited from C1.LiveLinq.LiveViews.View)

	GetEnumerator	Returns an enumerator that iterates through the view items. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
	GroupBy	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
	GroupJoin	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
	IndexOf	Searches for the specified object among the elements of the view and returns the zero-based ordinal position of its first occurrence. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
	Join	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
	Maintain	Overridden. This method overrides View.Maintain.
	OrderBy<TKey>	Sorts the elements of a view in ascending order. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
	OrderByDescending<TKey>	Sorts the elements of a view in descending order. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
	PurgeEmptyGroups	Overridden. This method overrides View.PurgeEmptyGroups.
	Rebuild	Overridden. This method overrides View.Rebuild.
	Select<TResult>	Projects each element of a view into a new form. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
	SelectMany	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
	SetTransaction	Sets the value of the Transaction property. (Inherited from

		C1.LiveLinq.LiveViews.View)
	ToString	Returns a string that represents this instance of GroupView<TKey,TElement>
	Union	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
	Where	Filters the source view based on a predicate. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)

Top

Public Events

	Name	Description
	Changed	Occurs after an item of the view or the entire view has changed. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)

Top

See Also

Reference

GroupView<TKey,TElement> Class
C1.LiveLinq.LiveViews Namespace

Methods
>

Name	Description
AsCollectionViewFactory	Returns an instance of System.ComponentModel.ICollectionViewFactory that can be used as a source of a System.Windows.Data.CollectionViewSource. (Inherited from C1.LiveLinq.LiveViews.View)
AsDynamic	Used for views with anonymous type constructor as the result selector, converts the View to a View<dynamic> so it can be used

	for data binding and programmatic access. (Inherited from C1.LiveLinq.LiveViews.View)
AttachAggregationView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
AttachView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
Concat	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
Contains	Determines whether the view contains a specified item. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
DeferMaintenance	Enters a defer cycle that you can use to make bulk changes to the view sources and delay automatic view maintenance. (Inherited from C1.LiveLinq.LiveViews.View)
GetEnumerator	Returns an enumerator that iterates through the view items. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
GroupBy	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
GroupJoin	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
IndexOf	Searches for the specified object among the elements of the view and returns the zero-based ordinal position of its first occurrence. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
Join	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
Maintain	Overridden. This method overrides View.Maintain.
OrderBy<TKey>	Sorts the elements of a view in ascending order. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
OrderByDescending<TKey>	Sorts the elements of a view in descending order. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)

PurgeEmptyGroups	Overridden. This method overrides View.PurgeEmptyGroups.
Rebuild	Overridden. This method overrides View.Rebuild.
Select<TResult>	Projects each element of a view into a new form. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
SelectMany	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
SetTransaction	Sets the value of the Transaction property. (Inherited from C1.LiveLinq.LiveViews.View)
ToString	Returns a string that represents this instance of GroupView<TKey,TElement>
Union	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
Where	Filters the source view based on a predicate. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)

Top

See Also

Reference

GroupView<TKey,TElement> Class
C1.LiveLinq.LiveViews Namespace

Maintain Method
This method overrides View.Maintain.

Syntax

Visual Basic (Declaration)
`Public Overrides NotOverridable Sub Maintain()`
C#
`public override void Maintain()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

GroupView<TKey,TElement> Class
GroupView<TKey,TElement> Members

PurgeEmptyGroups Method
This method overrides View.PurgeEmptyGroups.

Syntax

Visual Basic (Declaration)
`Public Overrides NotOverridable Sub PurgeEmptyGroups()`
C#
`public override void PurgeEmptyGroups()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

GroupView<TKey,TElement> Class
GroupView<TKey,TElement> Members

Rebuild Method
This method overrides View.Rebuild.

Syntax

Visual Basic (Declaration)

```
Public Overrides NotOverridable Sub Rebuild()
```

C#

```
public override void Rebuild()
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

GroupView<TKey,TElement> Class
GroupView<TKey,TElement> Members

ToString Method
Returns a string that represents this instance of GroupView<TKey,TElement>

Syntax

Visual Basic (Declaration)

```
Public Overrides Function ToString() As String
```

C#

```
public override string ToString()
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

GroupView<TKey,TElement> Class
GroupView<TKey,TElement> Members

Properties

For a list of all members of this type, see GroupView<TKey,TElement> members.

Public Properties

	Name	Description
	Count	Gets the total number of elements in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	CurrentItem	Gets the current item in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	DeferredMaintenance	Overridden. This property overrides DeferredMaintenance.
	IsReadOnly	Gets a value indicating whether this view is read-only, not updatable. (Inherited from C1.LiveLinq.LiveViews.View)
	Item	Gets the view item (element) at the specified ordinal position. (Inherited from C1.LiveLinq.LiveViews.View<TElement>)
	Key	Gets the key value of the group.
	MaintenanceMode	Overridden. This property overrides MaintenanceMode.
	MoveToFirstOnReset	Gets or sets a value indicating that the first item must be made current after initial loading or reset (on any C1.LiveLinq.SourceChangeType.Reset notification) if current item was not set by other means. The default is True. (Inherited from C1.LiveLinq.LiveViews.View)
	Order	Gets a value indicating whether and how this view preserves item order if it exists in its base data source. (Inherited from C1.LiveLinq.LiveViews.View)
	Parent	Gets the grouping view (the result of a **GroupBy** operation) to which

		this group belongs.
	Transaction	Gets an instance of C1.LiveLinq.ITransaction associated with the view. If a view has a transaction associated with it, that transaction's scope is opened automatically every time the view is updated, so the programmer does not need to do it manually in code. (Inherited from C1.LiveLinq.LiveViews.View)

Top

See Also

Reference

GroupView<TKey,TElement> Class
C1.LiveLinq.LiveViews Namespace

DeferredMaintenance Property
This property overrides DeferredMaintenance.

Syntax

Visual Basic (Declaration)

```
Public Overrides NotOverridable ReadOnly Property DeferredMaintenance As Boolean
```

C#

```
public override bool DeferredMaintenance {get;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

GroupView<TKey,TElement> Class
GroupView<TKey,TElement> Members

Key Property

Gets the key value of the group.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Key As TKey`
C#
`public TKey Key {get;}`

Remarks

The key value is common to the elements of this group.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

GroupView<TKey,TElement> Class
GroupView<TKey,TElement> Members

MaintenanceMode Property

This property overrides MaintenanceMode.

Syntax

Visual Basic (Declaration)
`Public Overrides NotOverridable Property MaintenanceMode As ViewMaintenanceMode`
C#
`public override ViewMaintenanceMode MaintenanceMode {get; set;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

GroupView<TKey,TElement> Class
GroupView<TKey,TElement> Members

Parent Property
Gets the grouping view (the result of a **GroupBy** operation) to which this group belongs.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Parent As View`
C#
`public View Parent {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

GroupView<TKey,TElement> Class
GroupView<TKey,TElement> Members

OrderedView<T>
The type of the elements in the view.

Represents a sorted view.

Object Model

Syntax

Visual Basic (Declaration)	

`Public Class OrderedView(Of T)`
`  Inherits View(Of T)`
`  Implements C1.LiveLinq.IObservableSource(Of T)`	

C#	
`public class OrderedView<T> : View<T>, C1.LiveLinq.IObservableSource<T>`	

Type Parameters

T

The type of the elements in the view.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.View
    C1.LiveLinq.LiveViews.View<T>
      C1.LiveLinq.LiveViews.OrderedView<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

OrderedView<T> Members
C1.LiveLinq.LiveViews Namespace
OrderBy<TKey>(Expression<Func<T,TKey>>) Method
OrderByDescending<TKey> Method

Overview
The type of the elements in the view.

Represents a sorted view.

Object Model

Syntax

Visual Basic (Declaration)

`Public Class OrderedView(Of T)`
`  Inherits View(Of T)`
`  Implements C1.LiveLinq.IObservableSource(Of T)`

C#
`public class OrderedView<T> : View<T>, C1.LiveLinq.IObservableSource<T>`

Type Parameters

T

The type of the elements in the view.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.View
    C1.LiveLinq.LiveViews.View<T>
      C1.LiveLinq.LiveViews.OrderedView<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

OrderedView<T> Members
C1.LiveLinq.LiveViews Namespace
OrderBy<TKey>(Expression<Func<T,TKey>>) Method
OrderByDescending<TKey> Method

Members

Properties Methods Events

The following tables list the members exposed by OrderedView<T>.

Public Properties

	Name	Description
	Count	Gets the total number of elements in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	CurrentItem	Gets the current item in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	DeferredMaintenance	Gets the effective value of MaintenanceMode. (Inherited from C1.LiveLinq.LiveViews.View)
	IsReadOnly	Gets a value indicating whether this view is read-only, not updatable. (Inherited from C1.LiveLinq.LiveViews.View)
	Item	Gets the view item (element) at the specified ordinal position. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	MaintenanceMode	Gets or sets a value controlling how the view is synchronized with changes in its base data. (Inherited from C1.LiveLinq.LiveViews.View)
	MoveToFirstOnReset	Gets or sets a value indicating that the first item must be made current after initial loading or reset (on any C1.LiveLinq.SourceChangeType.Reset notification) if current item was not set by other means. The default is True. (Inherited from C1.LiveLinq.LiveViews.View)
	Order	Gets a value indicating whether and how this view preserves item order if it exists in its base data source. (Inherited from C1.LiveLinq.LiveViews.View)

	Transaction	Gets an instance of C1.LiveLinq.ITransaction associated with the view. If a view has a transaction associated with it, that transaction's scope is opened automatically every time the view is updated, so the programmer does not need to do it manually in code. (Inherited from C1.LiveLinq.LiveViews.View)

Top

Public Methods

	Name	Description
	AsCollectionViewFactory	Returns an instance of System.ComponentModel.ICollectionViewFactory that can be used as a source of a System.Windows.Data.CollectionViewSource. (Inherited from C1.LiveLinq.LiveViews.View)
	AsDynamic	Used for views with anonymous type constructor as the result selector, converts the View to a View<dynamic> so it can be used for data binding and programmatic access. (Inherited from C1.LiveLinq.LiveViews.View)
	AttachAggregationView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	AttachView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Concat	Overloaded. Concatenation of two views. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Contains	Determines whether the view contains a specified item. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	DeferMaintenance	Enters a defer cycle that you can use to make bulk changes to the view sources and delay automatic view maintenance. (Inherited from C1.LiveLinq.LiveViews.View)

	GetEnumerator	Returns an enumerator that iterates through the view items. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	GroupBy	Overloaded. Groups the elements of a view according to a specified key selector function. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	GroupJoin	Overloaded. Correlates the elements of two views based on equality of keys and groups the results. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	IndexOf	Searches for the specified object among the elements of the view and returns the zero-based ordinal position of its first occurrence. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Join	Overloaded. Correlates the elements of two views based on matching keys. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Maintain	Brings the view up to date with its source data. (Inherited from C1.LiveLinq.LiveViews.View)
	OrderBy<TKey>	Sorts the elements of a view in ascending order. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	OrderByDescending<TKey>	Sorts the elements of a view in descending order. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	PurgeEmptyGroups	Remove empty groups from a grouping view. (Inherited from C1.LiveLinq.LiveViews.View)
	Rebuild	Re-populates the view by re-executing the view's query. (Inherited from C1.LiveLinq.LiveViews.View)
	Select<TResult>	Projects each element of a view into a new form. (Inherited from

		C1.LiveLinq.LiveViews.View<T>)
	SelectMany	Overloaded. Projects each element of this view to a collection of collections, flattens the resulting collections into one collection, and invokes a result selector function on each element therein. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	SetTransaction	Sets the value of the Transaction property. (Inherited from C1.LiveLinq.LiveViews.View)
	ThenBy<TKey>	Performs a subsequent ordering of view elements in ascending order according to a key.
	ThenByDescending<TKey>	Performs a subsequent ordering of view elements in descending order according to a key.
	ToString	Returns a string representing this view. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Union	Overloaded. Set union of two views. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Where	Filters the source view based on a predicate. (Inherited from C1.LiveLinq.LiveViews.View<T>)

Top

Public Events

	Name	Description
	Changed	Occurs after an item of the view or the entire view has changed. (Inherited from C1.LiveLinq.LiveViews.View<T>)

Top

See Also

Reference

OrderedView<T> Class
C1.LiveLinq.LiveViews Namespace
OrderBy<TKey>(Expression<Func<T,TKey>>) Method
OrderByDescending<TKey> Method

Methods
For a list of all members of this type, see OrderedView<T> members.

Public Methods

	Name	Description
	AsCollectionViewFactory	Returns an instance of System.ComponentModel.ICollectionViewFactory that can be used as a source of a System.Windows.Data.CollectionViewSource. (Inherited from C1.LiveLinq.LiveViews.View)
	AsDynamic	Used for views with anonymous type constructor as the result selector, converts the View to a View<dynamic> so it can be used for data binding and programmatic access. (Inherited from C1.LiveLinq.LiveViews.View)
	AttachAggregationView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	AttachView	Overloaded. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Concat	Overloaded. Concatenation of two views. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Contains	Determines whether the view contains a specified item. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	DeferMaintenance	Enters a defer cycle that you can use to make bulk changes to the view sources and delay automatic view maintenance. (Inherited from C1.LiveLinq.LiveViews.View)

	GetEnumerator	Returns an enumerator that iterates through the view items. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	GroupBy	Overloaded. Groups the elements of a view according to a specified key selector function. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	GroupJoin	Overloaded. Correlates the elements of two views based on equality of keys and groups the results. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	IndexOf	Searches for the specified object among the elements of the view and returns the zero-based ordinal position of its first occurrence. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Join	Overloaded. Correlates the elements of two views based on matching keys. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Maintain	Brings the view up to date with its source data. (Inherited from C1.LiveLinq.LiveViews.View)
	OrderBy<TKey>	Sorts the elements of a view in ascending order. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	OrderByDescending<TKey>	Sorts the elements of a view in descending order. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	PurgeEmptyGroups	Remove empty groups from a grouping view. (Inherited from C1.LiveLinq.LiveViews.View)
	Rebuild	Re-populates the view by re-executing the view's query. (Inherited from C1.LiveLinq.LiveViews.View)
	Select<TResult>	Projects each element of a view into a new form. (Inherited from

		C1.LiveLinq.LiveViews.View<T>)
	SelectMany	Overloaded. Projects each element of this view to a collection of collections, flattens the resulting collections into one collection, and invokes a result selector function on each element therein. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	SetTransaction	Sets the value of the Transaction property. (Inherited from C1.LiveLinq.LiveViews.View)
	ThenBy<TKey>	Performs a subsequent ordering of view elements in ascending order according to a key.
	ThenByDescending<TKey>	Performs a subsequent ordering of view elements in descending order according to a key.
	ToString	Returns a string representing this view. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Union	Overloaded. Set union of two views. (Inherited from C1.LiveLinq.LiveViews.View<T>)
	Where	Filters the source view based on a predicate. (Inherited from C1.LiveLinq.LiveViews.View<T>)

Top

See Also

Reference

OrderedView<T> Class
C1.LiveLinq.LiveViews Namespace
OrderBy<TKey>(Expression<Func<T,TKey>>) Method
OrderByDescending<TKey> Method

ThenBy<TKey> Method
The type of the key returned by *keySelector*.

A function to extract a key from each element.

Performs a subsequent ordering of view elements in ascending order according to a key.

Syntax

Visual Basic (Declaration)

`Public Function ThenBy(Of TKey)( _`
`  ByVal keySelector As Expression(Of Func(Of T,TKey)) _`
`) As OrderedView(Of T)`

C#

`public OrderedView<T> ThenBy<TKey>(`
`  Expression<Func<T,TKey>> keySelector`
`)`

Parameters

keySelector

A function to extract a key from each element.

Type Parameters

TKey

The type of the key returned by *keySelector*.

Return Value

A view whose elements are sorted according to a key.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

OrderedView<T> Class
OrderedView<T> Members

The type of the key returned by *keySelector*.

A function to extract a key from each element.

Performs a subsequent ordering of view elements in descending order according to a key.

Syntax

Visual Basic (Declaration)

`Public Function ThenByDescending(Of TKey)( _`
`  ByVal keySelector As Expression(Of Func(Of T,TKey)) _`
`) As OrderedView(Of T)`

C#

`public OrderedView<T> ThenByDescending<TKey>(`
`  Expression<Func<T,TKey>> keySelector`
`)`

Parameters

keySelector

A function to extract a key from each element.

Type Parameters

TKey

The type of the key returned by *keySelector*.

Return Value

A view whose elements are sorted in descending order according to a key.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

OrderedView<T> Class
OrderedView<T> Members

PropertyIsNotVirtualException
Represents an exception that indicates that a property used in a result selector of a live view is not virtual.

Object Model

Syntax

Visual Basic (Declaration)
`Public Class PropertyIsNotVirtualException` `  Inherits System.Exception`
C#
`public class PropertyIsNotVirtualException : System.Exception`

Inheritance Hierarchy

System.Object
  System.Exception
    C1.LiveLinq.LiveViews.PropertyIsNotVirtualException

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

PropertyIsNotVirtualException Members
C1.LiveLinq.LiveViews Namespace

Overview
Represents an exception that indicates that a property used in a result selector of a live view is not virtual.

Object Model

PropertyIsNotVirtualException

Syntax

Visual Basic (Declaration)
`Public Class PropertyIsNotVirtualException` `  Inherits System.Exception`
C#
`public class PropertyIsNotVirtualException : System.Exception`

Inheritance Hierarchy

System.Object
  System.Exception
    C1.LiveLinq.LiveViews.PropertyIsNotVirtualException

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

PropertyIsNotVirtualException Members
C1.LiveLinq.LiveViews Namespace

Members
Properties Methods

The following tables list the members exposed by PropertyIsNotVirtualException.

Public Properties

	Name	Description

	Context	Lambda expression where a type with the non-virtual property is used.
	Data	(Inherited from System.Exception)
	HelpLink	(Inherited from System.Exception)
	InnerException	(Inherited from System.Exception)
	Message	Overridden. Gets a message that describes the current exception.
	Property	The non-virtual property.
	ResultType	The result type of the lambda expression.
	Source	(Inherited from System.Exception)
	StackTrace	(Inherited from System.Exception)
	TargetSite	(Inherited from System.Exception)

Top

Public Methods

	Name	Description
	GetBaseException	(Inherited from System.Exception)
	GetObjectData	(Inherited from System.Exception)
	GetType	(Inherited from System.Exception)
	ToString	(Inherited from System.Exception)

Top

See Also

Reference

PropertyIsNotVirtualException Class
C1.LiveLinq.LiveViews Namespace

Properties
>

Name	Description
Context	Lambda expression where a type with the non-virtual property is used.
Data	(Inherited from System.Exception)
HelpLink	(Inherited from System.Exception)
InnerException	(Inherited from System.Exception)
Message	Overridden. Gets a message that describes the current exception.
Property	The non-virtual property.
ResultType	The result type of the lambda expression.
Source	(Inherited from System.Exception)
StackTrace	(Inherited from System.Exception)
TargetSite	(Inherited from System.Exception)

Top

See Also

Reference

PropertyIsNotVirtualException Class
C1.LiveLinq.LiveViews Namespace

Context Property
Lambda expression where a type with the non-virtual property is used.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Context As LambdaExpression`
C#
`public LambdaExpression Context {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

PropertyIsNotVirtualException Class
PropertyIsNotVirtualException Members

Message Property
Gets a message that describes the current exception.

Syntax

Visual Basic (Declaration)
`Public Overrides ReadOnly Property Message As String`
C#
`public override string Message {get;}`

Property Value

The error message that explains the reason for the exception.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

PropertyIsNotVirtualException Class
PropertyIsNotVirtualException Members

Property Property
The non-virtual property.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Property As PropertyInfo`
C#
`public PropertyInfo Property {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

PropertyIsNotVirtualException Class
PropertyIsNotVirtualException Members

ResultType Property
The result type of the lambda expression.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property ResultType As Type`
C#
`public Type ResultType {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

PropertyIsNotVirtualException Class
PropertyIsNotVirtualException Members

View
Base class for the View<T> class. Contains members that don't depend on the element type *T*.

Object Model

Syntax

Visual Basic (Declaration)
`Public MustInherit Class View`
C#
`public abstract class View`

Remarks

Use this class to type variables that can accept views with different element types or a view with anonymous element type.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.View
    C1.LiveLinq.LiveViews.View<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View Members
C1.LiveLinq.LiveViews Namespace

Overview
Base class for the View<T> class. Contains members that don't depend on the element type *T*.

Object Model

Syntax

Visual Basic (Declaration)
`Public MustInherit Class View`
C#
`public abstract class View`

Remarks

Use this class to type variables that can accept views with different element types or a view with anonymous element type.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.View
    C1.LiveLinq.LiveViews.View<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View Members
C1.LiveLinq.LiveViews Namespace

Members
Properties Methods

The following tables list the members exposed by View.

Public Properties

	Name	Description
	Count	Gets the total number of elements in the view.
	CurrentItem	Gets the current item in the view.
	DeferredMaintenance	Gets the effective value of MaintenanceMode.
	IsReadOnly	Gets a value indicating whether this view is read-only, not updatable.
	MaintenanceMode	Gets or sets a value controlling how the view is synchronized with changes in its base data.
	MoveToFirstOnReset	Gets or sets a value indicating that the first item must be made current after initial loading or reset (on any C1.LiveLinq.SourceChangeType.Reset notification) if current item was not set by other means. The default is True.
	Order	Gets a value indicating whether and how this view preserves item order if it exists in its base data source.
	Rows	Gets the collection of ViewRow objects used for programmatic access to view elements (items) and for data binding.
	Transaction	Gets an instance of C1.LiveLinq.ITransaction associated with the view. If a view has a transaction associated with it, that transaction's scope is

		opened automatically every time the view is updated, so the programmer does not need to do it manually in code.

Top

Public Methods

	Name	Description
S	AllowInResult	Specifies that a type with non-virtual properties can be used in a result selector.
	AsCollectionViewFactory	Returns an instance of System.ComponentModel.ICollectionViewFactory that can be used as a source of a System.Windows.Data.CollectionViewSource.
	AsDynamic	Used for views with anonymous type constructor as the result selector, converts the View to a View<dynamic> so it can be used for data binding and programmatic access.
	DeferMaintenance	Enters a defer cycle that you can use to make bulk changes to the view sources and delay automatic view maintenance.
	Maintain	Brings the view up to date with its source data.
	PurgeEmptyGroups	Remove empty groups from a grouping view.
	Rebuild	Re-populates the view by re-executing the view's query.
	SetTransaction	Sets the value of the Transaction property.

Top

See Also

Reference

View Class
C1.LiveLinq.LiveViews Namespace

Methods
For a list of all members of this type, see View members.

Public Methods

	Name	Description
S	AllowInResult	Specifies that a type with non-virtual properties can be used in a result selector.
	AsCollectionViewFactory	Returns an instance of System.ComponentModel.ICollectionViewFactory that can be used as a source of a System.Windows.Data.CollectionViewSource.
	AsDynamic	Used for views with anonymous type constructor as the result selector, converts the View to a View<dynamic> so it can be used for data binding and programmatic access.
	DeferMaintenance	Enters a defer cycle that you can use to make bulk changes to the view sources and delay automatic view maintenance.
	Maintain	Brings the view up to date with its source data.
	PurgeEmptyGroups	Remove empty groups from a grouping view.
	Rebuild	Re-populates the view by re-executing the view's query.
	SetTransaction	Sets the value of the Transaction property.

Top

See Also

Reference

View Class
C1.LiveLinq.LiveViews Namespace

AllowInResult Method
The type that is allowed to be used.

Specifies that a type with non-virtual properties can be used in a result selector.

Syntax

Visual Basic (Declaration)

```
Public Shared Sub AllowInResult( _
   ByVal type As Type _
)
```

C#

```
public static void AllowInResult(
   Type type
)
```

Parameters

type

The type that is allowed to be used.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View Class
View Members

AsCollectionViewFactory Method
Returns an instance of System.ComponentModel.ICollectionViewFactory that can be used as a source of a System.Windows.Data.CollectionViewSource.

Syntax

Visual Basic (Declaration)

```
Public Function AsCollectionViewFactory() As ICollectionViewFactory
```

C#

```
public ICollectionViewFactory AsCollectionViewFactory()
```

Return Value

A factory that returns the same View as a System.ComponentModel.ICollectionView.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View Class
View Members

AsDynamic Method
Used for views with anonymous type constructor as the result selector, converts the View to a View<dynamic> so it can be used for data binding and programmatic access.

Syntax

Visual Basic (Declaration)

```
Public Function AsDynamic() As View(Of Object)
```

C#

```
public View<object> AsDynamic()
```

Return Value

A dynamic view.

Remarks

A view with anonymous type constructor as the result selector cannot be used for data binding or programmatic access without applying AsDynamic to it. An attempt to do so results in an exception. After applying AsDynamic, such view can be used for data binding and programmatic access without limitations.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View Class
View Members

DeferMaintenance Method
Enters a defer cycle that you can use to make bulk changes to the view sources and delay automatic view maintenance.

Syntax

Visual Basic (Declaration)
`Public Function DeferMaintenance() As IDisposable`
C#
`public IDisposable DeferMaintenance()`

Return Value

An System.IDisposable object that you can use to dispose of the calling object.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View Class
View Members

Maintain Method
Brings the view up to date with its source data.

Syntax

Visual Basic (Declaration)
`Public Overridable Sub Maintain()`
C#
`public virtual void Maintain()`

Remarks

If source data have not changed since the last time the view was maintained (updated), this method does nothing. It also does nothing if the view's MaintenanceMode is **Immediate**, because in that case the view is guaranteed to be in synch with its base data at all times. If the view is in deferred mode (its MaintenanceMode property returns **true**), the programmer can use the **Maintain** method to force updating the view.

Note that it is not necessary to call **Maintain** to make sure you get updated data from the view. The view is automatically updated every time you request data from it, if base data changed since the last request, regardless of the view's MaintenanceMode.

LiveLinq maintains views using optimized incremental algorithms, not simply re-populates them from scratch. It calculating the delta in the view from the delta in the base data. In most cases, it allows to propagate the change from base data to the view very fast.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View Class
View Members

PurgeEmptyGroups Method
Remove empty groups from a grouping view.

Syntax

Visual Basic (Declaration)
`Public Overridable Sub PurgeEmptyGroups()`
C#
`public virtual void PurgeEmptyGroups()`

Remarks

This method is used only for **GroupBy** (grouping) views, does nothing for views of other kinds.

When a grouping view is populated, it does not contain empty groups. But later, as a result of maintaining the grouping view, keeping it in synch with changes in its base data, some of the groups can become empty. If it is undesirable to have empty groups, you can call this method to delete them.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View Class
View Members

Rebuild Method
Re-populates the view by re-executing the view's query.

Syntax

Visual Basic (Declaration)

```
Public Overridable Sub Rebuild()
```

C#

```
public virtual void Rebuild()
```

Remarks

This method is rarely needed, because normally automatic incremental maintenance is faster than re-executing the query over the entire base data collection. However, if for some reason you need to re-populate it from scratch, that can be done with the **Rebuild** method.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View Class
View Members

SetTransaction Method
The new value for the the Transaction property.

Set this parameter to True to prevent exception if you have writable property paths and want to ignore them (but be aware that updates through property paths are not tracked by the transaction). The default is False.

Sets the value of the Transaction property.

Syntax

Visual Basic (Declaration)

```
Public Sub SetTransaction( _
   ByVal transaction As ITransaction, _
   Optional ByVal allowPropertyPaths As Boolean _
)
```

C#

```
public void SetTransaction(
   ITransaction transaction,
   bool allowPropertyPaths
)
```

Parameters

transaction

The new value for the the Transaction property.

allowPropertyPaths

Set this parameter to True to prevent exception if you have writable property paths and want to ignore them (but be aware that updates through property paths are not tracked by the transaction). The default is False.

Remarks

If a writable property path exists in the view element type (for example, `Order.Customer.City`), then an exception is thrown unless suppressed by setting allowPropertyPaths to True, because modifying properties using such paths cannot be supported by transactions. To prevent the exception, set the allowPropertyPaths parameter to True, but make sure you do not use such property paths in your code and don't have a two-way data binding to such property path in your controls. If you modify a property using such path in your code or through data binding, that modification happens outside the transaction scope.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View Class
View Members

Properties
For a list of all members of this type, see View members.

Public Properties

	Name	Description
	Count	Gets the total number of elements in the view.
	CurrentItem	Gets the current item in the view.
	DeferredMaintenance	Gets the effective value of MaintenanceMode.
	IsReadOnly	Gets a value indicating whether this view is read-only, not updatable.
	MaintenanceMode	Gets or sets a value controlling how the view is synchronized with changes in its base data.
	MoveToFirstOnReset	Gets or sets a value indicating that the first item must be made current after initial loading or reset (on any C1.LiveLinq.SourceChangeType.Reset notification) if current item was not set by other means. The default is True.
	Order	Gets a value indicating whether and how this view preserves item order if it exists in its base data source.
	Rows	Gets the collection of ViewRow objects used for programmatic access to view elements (items) and for data binding.
	Transaction	Gets an instance of C1.LiveLinq.ITransaction associated with the view. If a view has a transaction associated with it, that transaction's scope is opened automatically every time the view is updated, so the programmer does not need to do it manually in code.

Top

See Also

Reference

View Class
C1.LiveLinq.LiveViews Namespace

Count Property
Gets the total number of elements in the view.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Count As Integer`
C#
`public int Count {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View Class
View Members

CurrentItem Property
Gets the current item in the view.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property CurrentItem As Object`
C#
`public object CurrentItem {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View Class
View Members

DeferredMaintenance Property
Gets the effective value of MaintenanceMode.

Syntax

Visual Basic (Declaration)
`Public Overridable ReadOnly Property DeferredMaintenance As Boolean`
C#
`public virtual bool DeferredMaintenance {get;}`

Property Value

true if MaintenanceMode is set to **Deferred**, or if it is set to **Default** and no listeners are registered with this view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View Class
View Members

IsReadOnly Property
Gets a value indicating whether this view is read-only, not updatable.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property IsReadOnly As Boolean`
C#
`public bool IsReadOnly {get;}`

Remarks

Properties exposed by a view can be updatable or read-only. Updatable properties of a view can be modified directly in the view.

All properties of a read-only (not updatable) view are read-only. In an updatable view, properties directly corresponding to base data (source) properties are updatable, calculated properties are read-only. For example, in a view `from x in X select new { x.P, Q = x.Q + 1 }` P is updatable and Q is read-only.

Read-only properties of a view cannot be modified directly in the view, but they still reflect up-to-date values of the source, so the difference is often not critical, you can always modify corresponding property in the source, that will automatically change the property in the view.

Also, a read-only view cannot be cleared or its rows deleted or new rows added directly in the view (but all these actions can be performed on the source data collection and they will normally result in the corresponding changes of the view).

A **Join** view is read-only by default, but one of its two parts can be made updatable using the C1.LiveLinq.LiveViewExtensions.AsUpdatable<T> method.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View Class
View Members
AsUpdatable<T> Method

ViewRow Class
ViewRowState Enumeration

MaintenanceMode Property
Gets or sets a value controlling how the view is synchronized with changes in its base data.

Syntax

Visual Basic (Declaration)
`Public Overridable Property MaintenanceMode As ViewMaintenanceMode`
C#
`public virtual ViewMaintenanceMode MaintenanceMode {get; set;}`

Remarks

A view in **Default** mode (which is the default value for this property) is effectively in **Immediate** mode if it has a listener (for example, if a GUI control is bound to it); otherwise it is in **Deferred** mode. To find out whether the view is effectively in **Deferred** or in **Immediate** mode, you can use the DeferredMaintenance property.

If you set this property to **Deferred**, no listeners are allowed to register with this view. An attempt to register a listener will result in an exception.

See Also: View Maintenance Mode.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View Class
View Members
DeferredMaintenance Property

MoveToFirstOnReset Property

Gets or sets a value indicating that the first item must be made current after initial loading or reset (on any C1.LiveLinq.SourceChangeType.Reset notification) if current item was not set by other means. The default is True.

Syntax

Visual Basic (Declaration)
`Public Property MoveToFirstOnReset As Boolean`
C#
`public bool MoveToFirstOnReset {get; set;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View Class
View Members

Order Property

Gets a value indicating whether and how this view preserves item order if it exists in its base data source.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Order As ViewOrder`
C#
`public ViewOrder Order {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View Class
View Members

Rows Property
Gets the collection of ViewRow objects used for programmatic access to view elements (items) and for data binding.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Rows As ViewRowCollection`
C#
`public ViewRowCollection Rows {get;}`

Remarks

There can be a view with elements of any type *T*, as represented by the generic class View<T>. That type is not always known beforehand, so it is not always possible to access properties of view elements with strong-typed (early binding) code. Dynamic (untyped, late binding) access to view elements is provided by the ViewRowCollection owned by the view.

The collection of *view rows* (ViewRow objects) is always synchronized with the collection of view elements. ViewRow objects provide programmatic access to view elements and their properties.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View Class
View Members

Transaction Property
Gets an instance of C1.LiveLinq.ITransaction associated with the view. If a view has a transaction associated with it, that transaction's scope is opened automatically every time the view is updated, so the programmer does not need to do it manually in code.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Transaction As ITransaction`
C#
`public ITransaction Transaction {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

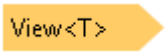
Reference

View Class
View Members

View<T>
The type of the elements in the view.

Represents a *live view*: a LINQ query result that supports two-way data binding and is kept up-to-date with base data.

Object Model



Syntax

Visual Basic (Declaration)	

`Public Class View(Of T)`
`  Inherits View`
`  Implements C1.LiveLinq.IObservableSource(Of T)`	

C#	
`public class View<T> : View, C1.LiveLinq.IObservableSource<T>`	

Type Parameters

T

The type of the elements in the view.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.View
    C1.LiveLinq.LiveViews.View<T>
      C1.Data.ClientView<T>
      C1.LiveLinq.LiveViews.AggregationView<TSource,TResult>
      C1.LiveLinq.LiveViews.GroupView<TKey,TElement>
      C1.LiveLinq.LiveViews.OrderedView<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

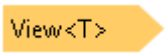
Reference

View<T> Members
C1.LiveLinq.LiveViews Namespace

Overview
The type of the elements in the view.

Represents a *live view*: a LINQ query result that supports two-way data binding and is kept up-to-date with base data.

Object Model



Syntax

Visual Basic (Declaration)

`Public Class View(Of T)`
`  Inherits View`
`  Implements C1.LiveLinq.IObservableSource(Of T)`

C#
`public class View<T> : View, C1.LiveLinq.IObservableSource<T>`

Type Parameters

T

The type of the elements in the view.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.View
    C1.LiveLinq.LiveViews.View<T>
      C1.Data.ClientView<T>
      C1.LiveLinq.LiveViews.AggregationView<TSource,TResult>
      C1.LiveLinq.LiveViews.GroupView<TKey,TElement>
      C1.LiveLinq.LiveViews.OrderedView<T>

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Members
C1.LiveLinq.LiveViews Namespace

Members
Properties Methods Events

The following tables list the members exposed by View<T>.

Public Properties

	Name	Description
	Count	Gets the total number of elements in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	CurrentItem	Gets the current item in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	DeferredMaintenance	Gets the effective value of MaintenanceMode. (Inherited from C1.LiveLinq.LiveViews.View)
	IsReadOnly	Gets a value indicating whether this view is read-only, not updatable. (Inherited from C1.LiveLinq.LiveViews.View)
	Item	Gets the view item (element) at the specified ordinal position.
	MaintenanceMode	Gets or sets a value controlling how the view is synchronized with changes in its base data. (Inherited from C1.LiveLinq.LiveViews.View)
	MoveToFirstOnReset	Gets or sets a value indicating that the first item must be made current after initial loading or reset (on any C1.LiveLinq.SourceChangeType.Reset notification) if current item was not set by other means. The default is True. (Inherited from C1.LiveLinq.LiveViews.View)
	Order	Gets a value indicating whether and how this view preserves item order if it exists in its base data source. (Inherited from

		C1.LiveLinq.LiveViews.View)
	Transaction	Gets an instance of C1.LiveLinq.ITransaction associated with the view. If a view has a transaction associated with it, that transaction's scope is opened automatically every time the view is updated, so the programmer does not need to do it manually in code. (Inherited from C1.LiveLinq.LiveViews.View)

Top

Public Methods

	Name	Description
	AsCollectionViewFactory	Returns an instance of System.ComponentModel.ICollectionViewFactory that can be used as a source of a System.Windows.Data.CollectionViewSource. (Inherited from C1.LiveLinq.LiveViews.View)
	AsDynamic	Used for views with anonymous type constructor as the result selector, converts the View to a View<dynamic> so it can be used for data binding and programmatic access. (Inherited from C1.LiveLinq.LiveViews.View)
	AttachAggregationView	Overloaded.
	AttachView	Overloaded.
	Concat	Overloaded. Concatenation of two views.
	Contains	Determines whether the view contains a specified item.
	DeferMaintenance	Enters a defer cycle that you can use to make bulk changes to the view sources and delay automatic view maintenance. (Inherited from C1.LiveLinq.LiveViews.View)

	GetEnumerator	Returns an enumerator that iterates through the view items.
	GroupBy	Overloaded. Groups the elements of a view according to a specified key selector function.
	GroupJoin	Overloaded. Correlates the elements of two views based on equality of keys and groups the results.
	IndexOf	Searches for the specified object among the elements of the view and returns the zero-based ordinal position of its first occurrence.
	Join	Overloaded. Correlates the elements of two views based on matching keys.
	Maintain	Brings the view up to date with its source data. (Inherited from C1.LiveLinq.LiveViews.View)
	OrderBy<TKey>	Sorts the elements of a view in ascending order.
	OrderByDescending<TKey>	Sorts the elements of a view in descending order.
	PurgeEmptyGroups	Remove empty groups from a grouping view. (Inherited from C1.LiveLinq.LiveViews.View)
	Rebuild	Re-populates the view by re-executing the view's query. (Inherited from C1.LiveLinq.LiveViews.View)
	Select<TResult>	Projects each element of a view into a new form.
	SelectMany	Overloaded. Projects each element of this view to a collection of collections, flattens the resulting collections into one collection, and invokes a result selector function on each element therein.
	SetTransaction	Sets the value of the Transaction property. (Inherited from

		C1.LiveLinq.LiveViews.View)
	ToString	Returns a string representing this view.
	Union	Overloaded. Set union of two views.
	Where	Filters the source view based on a predicate.

Top

Public Events

	Name	Description
	Changed	Occurs after an item of the view or the entire view has changed.

Top

See Also

Reference

View<T> Class
C1.LiveLinq.LiveViews Namespace

Methods
For a list of all members of this type, see View<T> members.

Public Methods

	Name	Description
	AsCollectionViewFactory	Returns an instance of System.ComponentModel.ICollectionViewFactory that can be used as a source of a System.Windows.Data.CollectionViewSource. (Inherited from C1.LiveLinq.LiveViews.View)
	AsDynamic	Used for views with anonymous type constructor as the result selector, converts the View to a View<dynamic> so it can be used

		for data binding and programmatic access. (Inherited from C1.LiveLinq.LiveViews.View)
	AttachAggregationView	Overloaded.
	AttachView	Overloaded.
	Concat	Overloaded. Concatenation of two views.
	Contains	Determines whether the view contains a specified item.
	DeferMaintenance	Enters a defer cycle that you can use to make bulk changes to the view sources and delay automatic view maintenance. (Inherited from C1.LiveLinq.LiveViews.View)
	GetEnumerator	Returns an enumerator that iterates through the view items.
	GroupBy	Overloaded. Groups the elements of a view according to a specified key selector function.
	GroupJoin	Overloaded. Correlates the elements of two views based on equality of keys and groups the results.
	IndexOf	Searches for the specified object among the elements of the view and returns the zero-based ordinal position of its first occurrence.
	Join	Overloaded. Correlates the elements of two views based on matching keys.
	Maintain	Brings the view up to date with its source data. (Inherited from C1.LiveLinq.LiveViews.View)
	OrderBy<TKey>	Sorts the elements of a view in ascending order.

	OrderByDescending<TKey>	Sorts the elements of a view in descending order.
	PurgeEmptyGroups	Remove empty groups from a grouping view. (Inherited from C1.LiveLinq.LiveViews.View)
	Rebuild	Re-populates the view by re-executing the view's query. (Inherited from C1.LiveLinq.LiveViews.View)
	Select<TResult>	Projects each element of a view into a new form.
	SelectMany	Overloaded. Projects each element of this view to a collection of collections, flattens the resulting collections into one collection, and invokes a result selector function on each element therein.
	SetTransaction	Sets the value of the Transaction property. (Inherited from C1.LiveLinq.LiveViews.View)
	ToString	Returns a string representing this view.
	Union	Overloaded. Set union of two views.
	Where	Filters the source view based on a predicate.

Top

See Also

Reference

View<T> Class
C1.LiveLinq.LiveViews Namespace

AttachAggregationView Method

Overload List

Overload	Description

AttachAggregationView<TResult>(Object,Func<View,AggregationView<T,TResult>>)	
AttachAggregationView<TResult>(Func<View,AggregationView<T,TResult>>)	

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members

AttachAggregationView<TResult>(Object,Func<View,AggregationView<T,TResult>>) Method

Syntax

Visual Basic (Declaration)

```
Public Overloads Function AttachAggregationView(Of TResult)( _
   ByVal subqueryId As Object, _
   ByVal selector As Func(Of View(Of T),AggregationView(Of T,TResult)) _
) As AggregationView(Of T,TResult)
```

C#

```
public AggregationView<T,TResult> AttachAggregationView<TResult>(
   object subqueryId,
   Func<View<T>,AggregationView<T,TResult>> selector
)
```

Parameters

subqueryId

selector

Type Parameters

TResult

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members
Overload List

AttachAggregationView<TResult>(Func<View,AggregationView<T,TResult>>) Method

Syntax

Visual Basic (Declaration)

```
Public Overloads Function AttachAggregationView(Of TResult)( _
   ByVal selector As Func(Of View(Of T),AggregationView(Of T,TResult)) _
) As AggregationView(Of T,TResult)
```

C#

```
public AggregationView<T,TResult> AttachAggregationView<TResult>(
   Func<View<T>,AggregationView<T,TResult>> selector
)
```

Parameters

selector

Type Parameters

TResult

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members
Overload List

AttachView Method

Overload List

Overload	Description
AttachView<TResult>(Func<View,View<TResult>>)	
AttachView<TResult>(Object,Func<View,View<TResult>>)	

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members

AttachView<TResult>(Func<View,View<TResult>>) Method

Syntax

Visual Basic (Declaration)

```
Public Overloads Function AttachView(Of TResult)( _
   ByVal selector As Func(Of View(Of T),View(Of TResult)) _
) As View(Of TResult)
```

C#

```
public View<TResult> AttachView<TResult>(
   Func<View<T>,View<TResult>> selector
)
```

Parameters

selector

Type Parameters

TResult

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members
Overload List

AttachView<TResult>(Object,Func<View,View<TResult>>) Method

Syntax

Visual Basic (Declaration)

```
Public Overloads Function AttachView(Of TResult)( _
   ByVal subqueryId As Object, _
   ByVal selector As Func(Of View(Of T),View(Of TResult)) _
) As View(Of TResult)
```

C#

```
public View<TResult> AttachView<TResult>(
   object subqueryId,
   Func<View<T>,View<TResult>> selector
)
```

Parameters

subqueryId

selector

Type Parameters

TResult

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members
Overload List

Concat Method
Concatenation of two views.

Overload List

Overload	Description
Concat(IObservableSource<T>)	Concatenation of two views.
Concat(ObservableCollection<T>)	Concatenation of this view and a collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members

Concat(IObservableSource<T>) Method
A collection (usually, a view) to concatenate to this view's collection of elements.

Concatenation of two views.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function Concat( _
   ByVal second As IObservableSource(Of T) _
) As View(Of T)
```

C#

```
public View<T> Concat(
   IObservableSource<T> second
)
```

Parameters

second

A collection (usually, a view) to concatenate to this view's collection of elements.

Return Value

The view that contains first the elements of this view and then the elements of the parameter collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members
Overload List

Concat(ObservableCollection<T>) Method
A collection to concatenate to this view's collection of elements.

Concatenation of this view and a collection.

Syntax

Visual Basic (Declaration)

`Public Overloads Function Concat( _`
`  ByVal second As ObservableCollection(Of T) _`
`) As View(Of T)`

C#

`public View<T> Concat(`
`  ObservableCollection<T> second`
`)`

Parameters

second

A collection to concatenate to this view's collection of elements.

Return Value

The view that contains first the elements of this view and then the elements of the parameter collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members
Overload List

Contains Method
The item to locate in the view.

Determines whether the view contains a specified item.

Syntax

Visual Basic (Declaration)

```
Public Function Contains( _
   ByVal item As T _
) As Boolean
```

C#

```
public bool Contains(
   T item
)
```

Parameters

item

The item to locate in the view.

Return Value

true if the view contains the specified item; otherwise, **false**.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members

GetEnumerator Method

Returns an enumerator that iterates through the view items.

Syntax

Visual Basic (Declaration)

```
Public Function GetEnumerator() As IEnumerator(Of T)
```

C#	
`public IEnumerator<T> GetEnumerator()`	

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members

GroupBy Method
Groups the elements of a view according to a specified key selector function.

Overload List

Overload	Description
GroupBy<TKey>(Expression<Func<T,TKey>>)	Groups the elements of a view according to a specified key selector function.

GroupBy<TKey,TElement>(Expression<Func<T,TKey>>,Expression<Func<T,TElement>>)	Groups the elements of a view according to a specified key selector function and projects the elements for each group by using a specified function.
GroupBy<TKey,TElement,TResult>(Expression<Func<T,TKey>>,Expression<Func<T,TElement>>,Expression<Func<TKey,IEnumerable<TElement>,TResult>>)	Groups the elements of a view

	according to a specified key selector function and creates a result value from each group and its key. The elements of each group are projected by using a specified function.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members

GroupBy<TKey>(Expression<Func<T,TKey>>) Method
The type of the key returned by *keySelector*.

A function to extract the key for each element.

Groups the elements of a view according to a specified key selector function.

Syntax

Visual Basic (Declaration)

`Public Overloads Function GroupBy(Of TKey)( _`
`  ByVal keySelector As Expression(Of Func(Of T,TKey)) _`
`) As View(Of GroupView(Of TKey,T))`

C#

`public View<GroupView<TKey,T>> GroupBy<TKey>(`
`  Expression<Func<T,TKey>> keySelector`
`)`

Parameters

keySelector

A function to extract the key for each element.

Type Parameters

TKey

The type of the key returned by *keySelector*.

Return Value

A view containing elements of type GroupView<TKey,TElement> each containing a key value and a view of the elements having that key value.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members
Overload List

GroupBy<TKey,TElement>(Expression<Func<T,TKey>>,Expression<Func<T,TElement>>) Method
The type of the key returned by *keySelector*.

The type of the element to which elements of each group are projected.

A function to extract the key for each element.

A function to map each source element to a *TElement*.

Groups the elements of a view according to a specified key selector function and projects the elements for each group by using a specified function.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function GroupBy
    (Of TKey,TElement)( _
   ByVal keySelector As Expression(Of Func(Of T,TKey)), _
   ByVal elementSelector As Expression(Of Func(Of T,TElement)) _
) As View(Of GroupView(Of TKey,TElement))
```

C#

```
public View<GroupView<TKey,TElement>> GroupBy<TKey,TElement>(
```

```
    Expression<Func<T,TKey>> keySelector,
    Expression<Func<T,TElement>> elementSelector
)
```

Parameters

keySelector

A function to extract the key for each element.

elementSelector

A function to map each source element to a *TElement*.

Type Parameters

TKey

The type of the key returned by *keySelector*.

TElement

The type of the element to which elements of each group are projected.

Return Value

A view containing elements of type GroupView<TKey,TElement> each containing a key value and a view of the elements projected (mapped) from the elements having that key value.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members
Overload List

GroupBy<TKey,TElement,TResult>(Expression<Func<T,TKey>>,Expression<Func<T,TElement>>,Expression<Func<TKey,IEnumerable<TElement>,TResult>>) Method

The type of the key returned by *keySelector*.

The type of the elements in groups.

The type of the result value returned by *resultSelector*.

A function to extract the key for each element.

A function to map each source element to an element in the IGrouping.

A function to create a result value from each group.

Groups the elements of a view according to a specified key selector function and creates a result value from each group and its key. The elements of each group are projected by using a specified function.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function GroupBy
    (Of TKey,TElement,TResult)( _
   ByVal keySelector As Expression(Of Func(Of T,TKey)), _
   ByVal elementSelector As Expression(Of Func(Of T,TElement)), _
   ByVal resultSelector As Expression(Of Func(Of TKey,IEnumerable(Of
TElement),TResult)) _
) As View(Of TResult)
```

C#

```
public View<TResult> GroupBy<TKey,TElement,TResult>(
   Expression<Func<T,TKey>> keySelector,
   Expression<Func<T,TElement>> elementSelector,
   Expression<Func<TKey,IEnumerable<TElement>,TResult>> resultSelector
)
```

Parameters

keySelector

A function to extract the key for each element.

elementSelector

A function to map each source element to an element in the IGrouping.

resultSelector

A function to create a result value from each group.

Type Parameters

TKey

The type of the key returned by *keySelector*.

TElement

The type of the elements in groups.

TResult

The type of the result value returned by *resultSelector*.

Return Value

A view of elements of type *TResult* where each element represents a projection over a group and its key.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members
Overload List

GroupJoin Method
Correlates the elements of two views based on equality of keys and groups the results.

Overload List

Overload	Description

GroupJoin<TInner,TKey,TResult>(IObservableSource<TInner>,Expression<Func<T,TKey>>,Expression<Func<TInner,TKey>>,Expression<Func<T,GroupView<TKey,TInner>,TResult>>)	Correlates the elements of two views based on equality of keys and groups the results.
GroupJoin<TInner,TKey,TResult>(ObservableCollection<TInner>,Expression<Func<T,TKey>>,Expression<Func<TInner,TKey>>,Expression<Func<T,GroupView<TKey,TInner>,TResult>>)	Correlates the elements of this view and a collection based on equal

	ity of keys and groups the results.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members

GroupJoin<TInner,TKey,TResult>(IObservableSource<TInner>,Expression<Func<T,TKey>>,Expression<Func<TInner,TKey>>,Expression<Func<T,GroupView<TKey,TInner>,TResult>>) Method
The type of the elements of the view to join with this view.

The type of the keys returned by the key selector functions.

The type of the result elements.

The collection (usually, a view) to join to this view.

A function to extract the join key from each element of this view.

A function to extract the join key from each element of the second view.

A function to create a result view from an element from this view and a collection of matching elements from the second view.

Correlates the elements of two views based on equality of keys and groups the results.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function GroupJoin
    (Of TInner,TKey,TResult)( _
   ByVal inner As IObservableSource(Of TInner), _
   ByVal outerKeySelector As Expression(Of Func(Of T,TKey)), _
   ByVal innerKeySelector As Expression(Of Func(Of TInner,TKey)), _
   ByVal resultSelector As Expression(Of Func(Of T,GroupView(Of
TKey,TInner),TResult)) _
) As View(Of TResult)
```

C#

```
public View<TResult> GroupJoin<TInner,TKey,TResult>(
   IObservableSource<TInner> inner,
   Expression<Func<T,TKey>> outerKeySelector,
   Expression<Func<TInner,TKey>> innerKeySelector,
   Expression<Func<T,GroupView<TKey,TInner>,TResult>> resultSelector
)
```

Parameters

inner

The collection (usually, a view) to join to this view.

outerKeySelector

A function to extract the join key from each element of this view.

innerKeySelector

A function to extract the join key from each element of the second view.

resultSelector

A function to create a result view from an element from this view and a collection of matching elements from the second view.

Type Parameters

TInner

The type of the elements of the view to join with this view.

TKey

The type of the keys returned by the key selector functions.

TResult

The type of the result elements.

Return Value

A view containing elements of type *TResult* that are obtained by performing a grouped join on two views.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members
Overload List

GroupJoin<TInner,TKey,TResult>(ObservableCollection<TInner>,Expression<Func<T,TKey>>,Expression<Func<TInner,TKey>>,Expression<Func<T,GroupView<TKey,TInner>,TResult>>) Method
The type of the elements of the collection to join with this view.

The type of the keys returned by the key selector functions.

The type of the result elements.

The collection to join to this view.

A function to extract the join key from each element of this view.

A function to extract the join key from each element of the collection.

A function to create a result view from an element from this view and a collection of matching elements from the second view.

Correlates the elements of this view and a collection based on equality of keys and groups the results.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function GroupJoin
    (Of TInner,TKey,TResult)( _
   ByVal inner As ObservableCollection(Of TInner), _
   ByVal outerKeySelector As Expression(Of Func(Of T,TKey)), _
   ByVal innerKeySelector As Expression(Of Func(Of TInner,TKey)), _
   ByVal resultSelector As Expression(Of Func(Of T,GroupView(Of
TKey,TInner),TResult)) _
) As View(Of TResult)
```

C#

```
public View<TResult> GroupJoin<TInner,TKey,TResult>(
   ObservableCollection<TInner> inner,
   Expression<Func<T,TKey>> outerKeySelector,
   Expression<Func<TInner,TKey>> innerKeySelector,
   Expression<Func<T,GroupView<TKey,TInner>,TResult>> resultSelector
)
```

Parameters

inner

The collection to join to this view.

outerKeySelector

A function to extract the join key from each element of this view.

innerKeySelector

A function to extract the join key from each element of the collection.

resultSelector

A function to create a result view from an element from this view and a collection of matching elements from the second view.

Type Parameters

TInner

The type of the elements of the collection to join with this view.

TKey

The type of the keys returned by the key selector functions.

TResult

The type of the result elements.

Return Value

A view containing elements of type *TResult* that are obtained by performing a grouped join on this view and the collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members
Overload List

IndexOf Method

The object to locate in the view.

Searches for the specified object among the elements of the view and returns the zero-based ordinal position of its first occurrence.

Syntax

Visual Basic (Declaration)

```
Public Function IndexOf( _
   ByVal item As T _
) As Integer
```

C#

```
public int IndexOf(
   T item
```

)

Parameters

item

The object to locate in the view.

Return Value

The zero-based ordinal position of the first occurrence of *item* in the view, if found; otherwise, -1.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members

Join Method
Correlates the elements of two views based on matching keys.

Overload List

Overload	Description
Join<TInner,TKey,TResult>(IObservableSource<TInner>,Expression<Func<T,TKey>>,Expression<Func<TInner,TKey>>,Expression<Func<T,TInner,TResult>>)	Correlates the elements of two views based

	on matching keys.
Join<TInner,TKey,TResult>(ObservableCollection<TInner>,Expression<Func<T,TKey>>,Expression<Func<TInner,TKey>>,Expression<Func<T,TInner,TResult>>)	Correlates the elements of this view and a collection based on matching keys.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members

Join<TInner,TKey,TResult>(IObservableSource<TInner>,Expression<Func<T,TKey>>,Expression<Func<TInner,TKey>>,Expression<Func<T,TInner,TResult>>) Method
The type of the elements of the view to join with this view.

The type of the keys returned by the key selector functions.

The type of the result elements.

The collection (usually, a view) to join to this view.

A function to extract the join key from each element of this view.

A function to extract the join key from each element of the second view.

A function to create a result element from two matching elements.

Correlates the elements of two views based on matching keys.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function Join
    (Of TInner,TKey,TResult)( _
   ByVal inner As IObservableSource(Of TInner), _
   ByVal outerKeySelector As Expression(Of Func(Of T,TKey)), _
   ByVal innerKeySelector As Expression(Of Func(Of TInner,TKey)), _
   ByVal resultSelector As Expression(Of Func(Of T,TInner,TResult)) _
) As View(Of TResult)
```

C#

```
public View<TResult> Join<TInner,TKey,TResult>(
   IObservableSource<TInner> inner,
   Expression<Func<T,TKey>> outerKeySelector,
   Expression<Func<TInner,TKey>> innerKeySelector,
   Expression<Func<T,TInner,TResult>> resultSelector
)
```

Parameters

inner

The collection (usually, a view) to join to this view.

outerKeySelector

A function to extract the join key from each element of this view.

innerKeySelector

A function to extract the join key from each element of the second view.

resultSelector

A function to create a result element from two matching elements.

Type Parameters

TInner

The type of the elements of the view to join with this view.

TKey

The type of the keys returned by the key selector functions.

TResult

The type of the result elements.

Return Value

A view containing elements of type *TResult* that are obtained by performing an inner join on two views.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members
Overload List

Join<TInner,TKey,TResult>(ObservableCollection<TInner>,Expression<Func<T,TKey>>,Expression<Func<TInner,TKey>>,Expression<Func<T,TInner,TResult>>) Method

The type of the elements of the collection to join with this view.

The type of the keys returned by the key selector functions.

The type of the result elements.

The collection to join to this view.

A function to extract the join key from each element of this view.

A function to extract the join key from each element of the collection.

A function to create a result element from two matching elements.

Correlates the elements of this view and a collection based on matching keys.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function Join
    (Of TInner,TKey,TResult)( _
   ByVal inner As ObservableCollection(Of TInner), _
   ByVal outerKeySelector As Expression(Of Func(Of T,TKey)), _
   ByVal innerKeySelector As Expression(Of Func(Of TInner,TKey)), _
   ByVal resultSelector As Expression(Of Func(Of T,TInner,TResult)) _
) As View(Of TResult)
```

C#

```
public View<TResult> Join<TInner,TKey,TResult>(
   ObservableCollection<TInner> inner,
   Expression<Func<T,TKey>> outerKeySelector,
   Expression<Func<TInner,TKey>> innerKeySelector,
   Expression<Func<T,TInner,TResult>> resultSelector
)
```

Parameters

inner

The collection to join to this view.

outerKeySelector

A function to extract the join key from each element of this view.

innerKeySelector

A function to extract the join key from each element of the collection.

resultSelector

A function to create a result element from two matching elements.

Type Parameters

TInner

The type of the elements of the collection to join with this view.

TKey

The type of the keys returned by the key selector functions.

TResult

The type of the result elements.

Return Value

A view containing elements of type *TResult* that are obtained by performing an inner join on this view and the collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members
Overload List

OrderBy<TKey> Method
The type of the key returned by *keySelector*.

A function to extract a key from an element.

Sorts the elements of a view in ascending order.

Syntax

Visual Basic (Declaration)	
`Public Function OrderBy(Of TKey)( _`	

```
   ByVal keySelector As Expression(Of Func(Of T,TKey)) _
) As OrderedView(Of T)
```

C#

```
public OrderedView<T> OrderBy<TKey>(
   Expression<Func<T,TKey>> keySelector
)
```

Parameters

keySelector

A function to extract a key from an element.

Type Parameters

TKey

The type of the key returned by *keySelector*.

Return Value

A view whose elements are sorted according to a key.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members

OrderByDescending<TKey> Method
The type of the key returned by *keySelector*.

A function to extract a key from an element.

Sorts the elements of a view in descending order.

Syntax

Visual Basic (Declaration)

```
Public Function OrderByDescending(Of TKey)( _
   ByVal keySelector As Expression(Of Func(Of T,TKey)) _
) As OrderedView(Of T)
```

C#

```
public OrderedView<T> OrderByDescending<TKey>(
   Expression<Func<T,TKey>> keySelector
)
```

Parameters

keySelector

A function to extract a key from an element.

Type Parameters

TKey

The type of the key returned by *keySelector*.

Return Value

A view whose elements are sorted in descending order according to a key.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members

Select<TResult> Method
The type of the value returned by *selector*.

A transform function to apply to each element.

Projects each element of a view into a new form.

Syntax

Visual Basic (Declaration)

```
Public Function Select(Of TResult)( _
   ByVal selector As Expression(Of Func(Of T,TResult)) _
) As View(Of TResult)
```

C#

```
public View<TResult> Select<TResult>(
   Expression<Func<T,TResult>> selector
)
```

Parameters

selector

A transform function to apply to each element.

Type Parameters

TResult

The type of the value returned by *selector*.

Return Value

A view whose elements are the result of invoking the transform function on each element of this view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members

SelectMany Method

Projects each element of this view to a collection of collections, flattens the resulting collections into one collection, and invokes a result selector function on each element therein.

Overload List

Overload	Description
SelectMany<TCollection,TResult>(Expression<Func<T,IObservableSource<TCollection>>>,Expression<Func<T,TCollection,TResult>>)	Projects each element of this view to a collection of collections, flattens the resulting collections into one collection, and invokes a result selector function on

	each element therein.
SelectMany<TResult>(Expression<Func<T,IObservableSource<TResult>>>)	Projects each element of this view to a collection of *TResult* and flattens the resulting collections into one view.
SelectMany<TCollection,TResult>(Expression<Func<T,ObservableCollection<TCollection>>>,Expression<Func<T,TCollection,TResult>>)	Projects each element of this view to a

	collection of collections, flattens the resulting collections into one collection, and invokes a result selector function on each element therein.
SelectMany<TResult>(Expression<Func<T,ObservableCollection<TResult>>>)	Projects each element of this view to a collecti

	on of *TResult* and flattens the resulting collections into one view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members

SelectMany<TCollection,TResult>(Expression<Func<T,IObservableSource<TCollection>>>,Expression<Func<T,TCollection,TResult>>) Method
The type of the intermediate elements collected by *collectionSelector*.

The type of the elements of the resulting view.

A transform function to apply to each element of this view.

A transform function to apply to each element of the intermediate collection.

Projects each element of this view to a collection of collections, flattens the resulting collections into one collection, and invokes a result selector function on each element therein.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function SelectMany
    (Of TCollection,TResult)( _
   ByVal collectionSelector As Expression(Of Func(Of T,IObservableSource(Of
TCollection))), _
   ByVal resultSelector As Expression(Of Func(Of T,TCollection,TResult)) _
) As View(Of TResult)
```

C#

```
public View<TResult> SelectMany<TCollection,TResult>(
   Expression<Func<T,IObservableSource<TCollection>>> collectionSelector,
   Expression<Func<T,TCollection,TResult>> resultSelector
)
```

Parameters

collectionSelector

A transform function to apply to each element of this view.

resultSelector

A transform function to apply to each element of the intermediate collection.

Type Parameters

TCollection

The type of the intermediate elements collected by *collectionSelector*.

TResult

The type of the elements of the resulting view.

Return Value

A view whose elements are the result of invoking the one-to-many transform function *collectionSelector* on each element of this view and then mapping each of those collection elements and their corresponding source element to a result element.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members
Overload List

SelectMany<TResult>(Expression<Func<T,IObservableSource<TResult>>>) Method
The type of the elements of the resulting view.

A transform function to apply to each element.

Projects each element of this view to a collection of *TResult* and flattens the resulting collections into one view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function SelectMany(Of TResult)( _
   ByVal selector As Expression(Of Func(Of T,IObservableSource(Of TResult))) _
) As View(Of TResult)
```

C#

```
public View<TResult> SelectMany<TResult>(
   Expression<Func<T,IObservableSource<TResult>>> selector
)
```

Parameters

selector

A transform function to apply to each element.

Type Parameters

TResult

The type of the elements of the resulting view.

Return Value

A view whose elements are the result of invoking the one-to-many transform function on each element of this view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members
Overload List

SelectMany<TCollection,TResult>(Expression<Func<T,ObservableCollection<TCollection>>>,Expression<Func<T,TCollection,TResult>>) Method
The type of the intermediate elements collected by *collectionSelector*.

The type of the elements of the resulting view.

A transform function to apply to each element of this view.

A transform function to apply to each element of the intermediate collection.

Projects each element of this view to a collection of collections, flattens the resulting collections into one collection, and invokes a result selector function on each element therein.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function SelectMany
    (Of TCollection,TResult)( _
   ByVal collectionSelector As Expression(Of Func(Of T,ObservableCollection(Of
TCollection))), _
   ByVal resultSelector As Expression(Of Func(Of T,TCollection,TResult)) _
) As View(Of TResult)
```

C#

```
public View<TResult> SelectMany<TCollection,TResult>(
   Expression<Func<T,ObservableCollection<TCollection>>> collectionSelector,
   Expression<Func<T,TCollection,TResult>> resultSelector
)
```

Parameters

collectionSelector

A transform function to apply to each element of this view.

resultSelector

A transform function to apply to each element of the intermediate collection.

Type Parameters

TCollection

The type of the intermediate elements collected by *collectionSelector*.

TResult

The type of the elements of the resulting view.

Return Value

A view whose elements are the result of invoking the one-to-many transform function *collectionSelector* on each element of this view and then mapping each of those collection elements and their corresponding source element to a result element.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members
Overload List

The type of the elements of the resulting view.

A transform function to apply to each element.

Projects each element of this view to a collection of *TResult* and flattens the resulting collections into one view.

Syntax

Visual Basic (Declaration)

`Public Overloads Function SelectMany(Of TResult)( _`
`  ByVal selector As Expression(Of Func(Of T,ObservableCollection(Of TResult)))`
`_`
`) As View(Of TResult)`

C#

`public View<TResult> SelectMany<TResult>(`
`  Expression<Func<T,ObservableCollection<TResult>>> selector`
`)`

Parameters

selector

A transform function to apply to each element.

Type Parameters

TResult

The type of the elements of the resulting view.

Return Value

A view whose elements are the result of invoking the one-to-many transform function on each element of this view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members
Overload List

ToString Method
Returns a string representing this view.

Syntax

Visual Basic (Declaration)
`Public Overrides Function ToString() As String`
C#
`public override string ToString()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members

Union Method
Set union of two views.

Overload List

Overload	Description
Union(IObservableSource<T>)	Set union of two views.

Union(ObservableCollection<T>)	Set union of this view and a collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members

Union(IObservableSource<T>) Method
A collection (usually, a view) whose distinct elements form the second set for the union.

Set union of two views.

Syntax

Visual Basic (Declaration)

```
Public Overloads Function Union( _
   ByVal second As IObservableSource(Of T) _
) As View(Of T)
```

C#

```
public View<T> Union(
   IObservableSource<T> second
)
```

Parameters

second

A collection (usually, a view) whose distinct elements form the second set for the union.

Return Value

The view that contains the elements from both input views, excluding duplicates.

Remarks

This method excludes duplicates from the result set. This is different behavior to the Concat method, which returns all the elements in the input sequences including duplicates.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members
Overload List

Union(ObservableCollection<T>) Method
A collection whose distinct elements form the second set for the union.

Set union of this view and a collection.

Syntax

Visual Basic (Declaration)

`Public Overloads Function Union( _`
`  ByVal second As ObservableCollection(Of T) _`
`) As View(Of T)`

C#

`public View<T> Union(`
`  ObservableCollection<T> second`
`)`

Parameters

second

A collection whose distinct elements form the second set for the union.

Return Value

The view that contains the elements from both this view and the collection, excluding duplicates.

Remarks

This method excludes duplicates from the result set. This is different behavior to the Concat(ObservableCollection<T>) method, which returns all the elements in the input sequences including duplicates.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members
Overload List

Where Method

A function to test each element for a condition.

Filters the source view based on a predicate.

Syntax

Visual Basic (Declaration)

```
Public Function Where( _
   ByVal predicate As Expression(Of Func(Of T,Boolean)) _
) As View(Of T)
```

C#

```
public View<T> Where(
   Expression<Func<T,bool>> predicate
)
```

Parameters

predicate

A function to test each element for a condition.

Return Value

A view that contains elements of this view that satisfy the condition.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members

Properties
For a list of all members of this type, see View<T> members.

Public Properties

	Name	Description
	Count	Gets the total number of elements in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	CurrentItem	Gets the current item in the view. (Inherited from C1.LiveLinq.LiveViews.View)
	DeferredMaintenance	Gets the effective value of MaintenanceMode. (Inherited from C1.LiveLinq.LiveViews.View)
	IsReadOnly	Gets a value indicating whether this view is read-only, not updatable. (Inherited from C1.LiveLinq.LiveViews.View)
	Item	Gets the view item (element) at the specified ordinal position.

	MaintenanceMode	Gets or sets a value controlling how the view is synchronized with changes in its base data. (Inherited from C1.LiveLinq.LiveViews.View)
	MoveToFirstOnReset	Gets or sets a value indicating that the first item must be made current after initial loading or reset (on any C1.LiveLinq.SourceChangeType.Reset notification) if current item was not set by other means. The default is True. (Inherited from C1.LiveLinq.LiveViews.View)
	Order	Gets a value indicating whether and how this view preserves item order if it exists in its base data source. (Inherited from C1.LiveLinq.LiveViews.View)
	Transaction	Gets an instance of C1.LiveLinq.ITransaction associated with the view. If a view has a transaction associated with it, that transaction's scope is opened automatically every time the view is updated, so the programmer does not need to do it manually in code. (Inherited from C1.LiveLinq.LiveViews.View)

Top

See Also

Reference

View<T> Class
C1.LiveLinq.LiveViews Namespace

Item Property
The zero-based ordinal position of the view item.

Gets the view item (element) at the specified ordinal position.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Default Property Item( _
   ByVal ordinal As Integer _
```

```
) As T
```

C#

```
public T this[
   int ordinal
]; {get;}
```

Parameters

ordinal

The zero-based ordinal position of the view item.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

View<T> Class
View<T> Members

Events
For a list of all members of this type, see View<T> members.

Public Events

	Name	Description
	Changed	Occurs after an item of the view or the entire view has changed.

Top

See Also

Reference

View<T> Class
C1.LiveLinq.LiveViews Namespace

Changed Event

Occurs after an item of the view or the entire view has changed.

Syntax

Visual Basic (Declaration)
`Public Event Changed As EventHandler(Of SourceChangeEventArgs(Of T))`
C#
`public event EventHandler<SourceChangeEventArgs<T>> Changed`

Event Data

The event handler receives an argument of type SourceChangeEventArgs<T> containing data related to this event. The following **SourceChangeEventArgs<T>** properties provide information specific to this event.

Property	Description
ChangeType	Gets the type of change.
Item	Gets the object that is being changed.
Ordinal	Gets the ordinal position of the collection item that is being changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

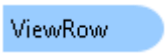
Reference

View<T> Class
View<T> Members

ViewRow

Represents a view element (item) for the purposes of dynamic, programmatic access to its properties and data binding.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit Class ViewRow`
C#
`public abstract class ViewRow`

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.ViewRow

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRow Members
C1.LiveLinq.LiveViews Namespace
Rows Property

Overview
Represents a view element (item) for the purposes of dynamic, programmatic access to its properties and data binding.

Object Model

Syntax

Visual Basic (Declaration)
`Public MustInherit Class ViewRow`
C#
`public abstract class ViewRow`

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.ViewRow

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRow Members
C1.LiveLinq.LiveViews Namespace
Rows Property

Members
Properties Methods Events

The following tables list the members exposed by ViewRow.

Public Properties

	Name	Description
	Item	Overloaded. Gets or sets a value of the specified view property.
	RowState	Gets the state of a view row with regard to edit, add and delete operations if they are performed directly on the view.
	Tag	Gets or sets user-supplied data associated with the view item.

	Value	Gets the view element (item) represented by this ViewRow object.
	View	Gets the view to which the ViewRow belongs.

Top

Public Methods

	Name	Description
	BeginEdit	Puts the ViewRow into edit mode.
	CancelEdit	Cancels the edit occurring on the row.
	Delete	Deletes a view item.
	EndEdit	Ends the edit occurring on the row.
	ToString	Gets the string representing this view row.

Top

Public Events

	Name	Description
	PropertyChanged	Occurs when a property value changes, after it has been changed.

Top

See Also

Reference

ViewRow Class
C1.LiveLinq.LiveViews Namespace
Rows Property

Methods
For a list of all members of this type, see ViewRow members.

Public Methods

	Name	Description
	BeginEdit	Puts the ViewRow into edit mode.
	CancelEdit	Cancels the edit occurring on the row.
	Delete	Deletes a view item.
	EndEdit	Ends the edit occurring on the row.
	ToString	Gets the string representing this view row.

Top

See Also

Reference

ViewRow Class
C1.LiveLinq.LiveViews Namespace
Rows Property

BeginEdit Method
Puts the ViewRow into edit mode.

Syntax

Visual Basic (Declaration)
`Public Sub BeginEdit()`
C#
`public void BeginEdit()`

Remarks

In edit mode, events and notifications are temporarily suspended, letting the user make changes to more than one property without triggering validation rules.

Edit mode is a standard feature of .NET data binding mechanism, supported by every data source implementing the System.ComponentModel.IEditableObject interface. For detailed explanation, see, for example, System.Data.DataRowView in .NET Framework documentation.

While a view item is in edit mode, changes made to its updatable properties directly in the view are not propagated to the corresponding base data properties until the edit operation is completed by a call to EndEdit (see View.IsReadOnly about updatability of view element properties directly in the view).

If you change base data (source) properties, those changes are propagated to this view and other views depending on that base data according to the normal view maintenance process, regardless of whether the view row is in edit mode or not.

Many-to-one relations between view properties are maintained automatically on changes made to updatable properties directly in the view, regardless of whether the view row is in edit mode or not. For example, if you set a CustomerID directly in the view, CustomerName will change accordingly, even if you do it in edit mode.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRow Class
ViewRow Members

CancelEdit Method
Cancels the edit occurring on the row.

Syntax

Visual Basic (Declaration)
`Public Overridable Sub CancelEdit()`
C#
`public virtual void CancelEdit()`

Remarks

If the user set some of the updatable properties of the row while it was in edit mode, the changes to those properties are rolled back and not propagated to the corresponding base data properties.

See the BeginEdit method for more information.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRow Class
ViewRow Members

Delete Method
Deletes a view item.

Syntax

Visual Basic (Declaration)
`Public Overridable Sub Delete()`
C#
`public virtual void Delete()`

Remarks

Deleting a view item is an update operation on the view. As any other update operation performed directly on the view (as opposed to on the base data collection on which that view depends, see C1.LiveLinq.LiveViewExtensions.AsUpdatable<T>), it is allowed only if the view is not read-only (see View.IsReadOnly), and it results in updating (in this case, deleting an item from) one of the view's base data collections.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRow Class
ViewRow Members

EndEdit Method
Ends the edit occurring on the row.

Syntax

Visual Basic (Declaration)
`Public Overridable Sub EndEdit()`
C#
`public virtual void EndEdit()`

Remarks

If the user set some of the updatable properties of the row while it was in edit mode, the changes to those properties are propagated to the corresponding base data properties at this point.

See the BeginEdit method for more information.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRow Class
ViewRow Members

ToString Method
Gets the string representing this view row.

Syntax

Visual Basic (Declaration)
`Public Overrides Function ToString() As String`
C#
`public override string ToString()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRow Class
ViewRow Members

Properties
For a list of all members of this type, see ViewRow members.

Public Properties

	Name	Description
	Item	Overloaded. Gets or sets a value of the specified view property.
	RowState	Gets the state of a view row with regard to edit, add and delete operations if they are performed directly on the view.
	Tag	Gets or sets user-supplied data associated with the view item.
	Value	Gets the view element (item) represented by this ViewRow object.
	View	Gets the view to which the ViewRow belongs.

Top

See Also

Reference

ViewRow Class
C1.LiveLinq.LiveViews Namespace
Rows Property

Item Property
Gets or sets a value of the specified view property.

Overload List

Overload	Description
Item(Int32)	Gets or sets a value of the specified view property.
Item(String)	Gets or sets a value of the specified view property.

Remarks

Setting a view property is an update operation on the view. As any other update operation performed directly on the view (as opposed to on the base data collection on which that view depends, see C1.LiveLinq.LiveViewExtensions.AsUpdatable<T>), it is allowed only if the view is not read-only (see View.IsReadOnly), and it results in updating one of the view's base data collections.

Only updatable properties can be set. An attempt to set a read-only property results in an exception. See View.IsReadOnly for more details.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRow Class
ViewRow Members

Item(Int32) Property
The ordinal position of the property in the collection of public properties of the type of the view element.

Gets or sets a value of the specified view property.

Syntax

Visual Basic (Declaration)

```
Public Overloads Property Item( _
   ByVal propertyOrdinal As Integer _
) As Object
```

C#

```
public object Item(
   int propertyOrdinal
) {get; set;}
```

Parameters

propertyOrdinal

The ordinal position of the property in the collection of public properties of the type of the view element.

Property Value

The value of the property.

Remarks

Setting a view property is an update operation on the view. As any other update operation performed directly on the view (as opposed to on the base data collection on which that view depends, see C1.LiveLinq.LiveViewExtensions.AsUpdatable<T>), it is allowed only if the view is not read-only (see View.IsReadOnly), and it results in updating one of the view's base data collections.

Only updatable properties can be set. An attempt to set a read-only property results in an exception. See View.IsReadOnly for more details.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRow Class
ViewRow Members
Overload List

Item(String) Property
The name of the property.

Gets or sets a value of the specified view property.

Syntax

Visual Basic (Declaration)

```
Public Overloads Property Item( _
   ByVal propertyName As String _
) As Object
```

C#

```
public object Item(
   string propertyName
) {get; set;}
```

Parameters

propertyName

The name of the property.

Property Value

The value of the property.

Remarks

Setting a view property is an update operation on the view. As any other update operation performed directly on the view (as opposed to on the base data collection on which that view depends, see C1.LiveLinq.LiveViewExtensions.AsUpdatable<T>), it is allowed only if the view is not read-only (see View.IsReadOnly), and it results in updating one of the view's base data collections.

Only updatable properties can be set. An attempt to set a read-only property results in an exception. See View.IsReadOnly for more details.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRow Class
ViewRow Members
Overload List

RowState Property
Gets the state of a view row with regard to edit, add and delete operations if they are performed directly on the view.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property RowState As ViewRowState`
C#
`public ViewRowState RowState {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRow Class
ViewRow Members

Tag Property
Gets or sets user-supplied data associated with the view item.

Syntax

Visual Basic (Declaration)
`Public Property Tag As Object`
C#
`public object Tag {get; set;}`

Remarks

Use this property to store any object you want to associate in your code with the view item that you need to access quickly.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRow Class
ViewRow Members

Value Property
Gets the view element (item) represented by this ViewRow object.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Value As Object`
C#
`public object Value {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRow Class
ViewRow Members

View Property
Gets the view to which the ViewRow belongs.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property View As View`
C#
`public View View {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRow Class
ViewRow Members

Events
For a list of all members of this type, see ViewRow members.

Public Events

	Name	Description

	PropertyChanged	Occurs when a property value changes, after it has been changed.

Top

See Also

Reference

ViewRow Class
C1.LiveLinq.LiveViews Namespace
Rows Property

PropertyChanged Event
Occurs when a property value changes, after it has been changed.

Syntax

Visual Basic (Declaration)
`Public Event PropertyChanged As PropertyChangedEventHandler`
C#
`public event PropertyChangedEventHandler PropertyChanged`

Event Data

The event handler receives an argument of type PropertyChangedEventArgs containing data related to this event. The following **PropertyChangedEventArgs** properties provide information specific to this event.

Property	Description
PropertyName	

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

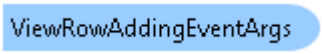
See Also

Reference

ViewRow Class
ViewRow Members

ViewRowAddingEventArgs
Provides data for the ViewRowCollection.ViewRowAdding event.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class ViewRowAddingEventArgs` `  Inherits System.EventArgs`
C#
`public class ViewRowAddingEventArgs : System.EventArgs`

Inheritance Hierarchy

System.Object
  System.EventArgs
    C1.LiveLinq.LiveViews.ViewRowAddingEventArgs

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

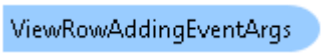
See Also

Reference

ViewRowAddingEventArgs Members
C1.LiveLinq.LiveViews Namespace

Overview
Provides data for the ViewRowCollection.ViewRowAdding event.

Object Model



Syntax

Visual Basic (Declaration)	
`Public Class ViewRowAddingEventArgs` `  Inherits System.EventArgs`	
C#	
`public class ViewRowAddingEventArgs : System.EventArgs`	

Inheritance Hierarchy

System.Object
  System.EventArgs
    C1.LiveLinq.LiveViews.ViewRowAddingEventArgs

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowAddingEventArgs Members
C1.LiveLinq.LiveViews Namespace

Members
Properties

The following tables list the members exposed by ViewRowAddingEventArgs.

Public Properties

	Name	Description

	Row	Gets the new view row that has just been added to ViewRowCollection.

Top

See Also

Reference

ViewRowAddingEventArgs Class
C1.LiveLinq.LiveViews Namespace

Properties
For a list of all members of this type, see ViewRowAddingEventArgs members.

Public Properties

	Name	Description
	Row	Gets the new view row that has just been added to ViewRowCollection.

Top

See Also

Reference

ViewRowAddingEventArgs Class
C1.LiveLinq.LiveViews Namespace

Row Property
Gets the new view row that has just been added to ViewRowCollection.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Row As ViewRow`
C#
`public ViewRow Row {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

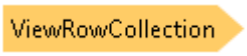
See Also

Reference

ViewRowAddingEventArgs Class
ViewRowAddingEventArgs Members

ViewRowCollection
Represents a collection of ViewRow objects used for programmatic access to view elements (items) and for data binding.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit Class ViewRowCollection` `  Implements C1.LiveLinq.IObservableSource(Of ViewRow)`
C#
`public abstract class ViewRowCollection : C1.LiveLinq.IObservableSource<ViewRow>`

Remarks

A **ViewRowCollection** is owned by a view, see View.Rows.

The collection of *view rows* (ViewRow objects) is always synchronized with the collection of view elements.

ViewRow objects provide programmatic access to view elements and their properties. Also, the View.Rows collection serves as the data source for data binding, when you bind a control or another client to a view.

Inheritance Hierarchy

System.Object
C1.LiveLinq.LiveViews.ViewRowCollection

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

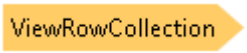
See Also

Reference

ViewRowCollection Members
C1.LiveLinq.LiveViews Namespace

Overview
Represents a collection of ViewRow objects used for programmatic access to view elements (items) and for data binding.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit Class ViewRowCollection` `  Implements C1.LiveLinq.IObservableSource(Of ViewRow)`
C#
`public abstract class ViewRowCollection : C1.LiveLinq.IObservableSource<ViewRow>`

Remarks

A **ViewRowCollection** is owned by a view, see View.Rows.

The collection of *view rows* (ViewRow objects) is always synchronized with the collection of view elements.

ViewRow objects provide programmatic access to view elements and their properties. Also, the View.Rows collection serves as the data source for data binding, when you bind a control or another client to a view.

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.ViewRowCollection

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Members
C1.LiveLinq.LiveViews Namespace

Members
Properties Methods Events

The following tables list the members exposed by ViewRowCollection.

Public Properties

	Name	Description
	AllowClear	Gets a value indicating whether the Clear operation is allowed on the view directly.
	AllowEdit	Gets a value indicating whether modifying property values is allowed on the view directly.
	AllowNew	Gets a value indicating whether the CreateRow operation is allowed on the view directly.
	AllowRemove	Gets a value indicating whether deleting rows is allowed on the view

		directly.
	Count	Gets the number of elements in the view.
	GroupDescriptions	Gets a collection of System.ComponentModel.GroupDescription objects that describe how the items in the collection are grouped.
	Groups	Gets the top-level groups.
	Item	Gets the view row at the specified ordinal position.
	Properties	Returns the collection of properties available in the view element type to programmatic access through ViewRow and to data binding.
	SortDescriptions	Gets a collection of System.ComponentModel.SortDescription objects that describe how the items in the collection are sorted.

Top

Public Methods

	Name	Description
	Clear	Deletes all view elements.
	Contains	Determines whether the ViewRowCollection contains a specific view row.
	CreateRow	Creates a new item directly in the view, and a view row associated with it, and adds it to the ViewRowCollection.
	DeferRefresh	Enters a defer cycle that you can use to merge changes to the view and delay automatic refresh.
	GetEnumerator	Returns an enumerator that iterates through the collection.
	IndexOf	Determines the ordinal position of a specific view row in the

		ViewRowCollection.
	Remove	Deletes the specified view item.
	RemoveAt	Deletes the view row at a specified ordinal position in ViewRowCollection.

Top

Public Events

	Name	Description
	Changed	Occurs after a view row has changed.
	ViewRowAdding	Occurs after a new view row is created so it can be populated with default values.

Top

See Also

Reference

ViewRowCollection Class
C1.LiveLinq.LiveViews Namespace

Methods
For a list of all members of this type, see ViewRowCollection members.

Public Methods

	Name	Description
	Clear	Deletes all view elements.
	Contains	Determines whether the ViewRowCollection contains a specific view row.
	CreateRow	Creates a new item directly in the view, and a view row associated with it, and adds it to the ViewRowCollection.

	DeferRefresh	Enters a defer cycle that you can use to merge changes to the view and delay automatic refresh.
	GetEnumerator	Returns an enumerator that iterates through the collection.
	IndexOf	Determines the ordinal position of a specific view row in the ViewRowCollection.
	Remove	Deletes the specified view item.
	RemoveAt	Deletes the view row at a specified ordinal position in ViewRowCollection.

Top

See Also

Reference

ViewRowCollection Class
C1.LiveLinq.LiveViews Namespace

Clear Method
Deletes all view elements.

Syntax

Visual Basic (Declaration)

```
Public Sub Clear()
```

C#

```
public void Clear()
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

Contains Method
The view row to locate in the collection.

Determines whether the ViewRowCollection contains a specific view row.

Syntax

Visual Basic (Declaration)

`Public Function Contains( _`
`  ByVal row As ViewRow _`
`) As Boolean`

C#

`public bool Contains(`
`  ViewRow row`
`)`

Parameters

row

The view row to locate in the collection.

Return Value

true if the view row is found in the collection; otherwise, **false**.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

CreateRow Method

Creates a new item directly in the view, and a view row associated with it, and adds it to the ViewRowCollection.

Syntax

Visual Basic (Declaration)
`Public Function CreateRow() As ViewRow`
C#
`public ViewRow CreateRow()`

Return Value

The newly created view row.

Remarks

Creating a new row is an update operation on the view. As any other update operation performed directly on the view (as opposed to on the base data collection on which that view depends, see C1.LiveLinq.LiveViewExtensions.AsUpdatable<T>), it is allowed only if the view is not read-only (see View.IsReadOnly), and it results in updating (in this case, adding a new item to) one of the view's base data collections.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

DeferRefresh Method

Enters a defer cycle that you can use to merge changes to the view and delay automatic refresh.

Syntax

Visual Basic (Declaration)
`Public Function DeferRefresh() As IDisposable`
C#
`public IDisposable DeferRefresh()`

Return Value

An System.IDisposable object that you can use to dispose of the calling object.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

GetEnumerator Method
Returns an enumerator that iterates through the collection.

Syntax

Visual Basic (Declaration)
`Public Function GetEnumerator() As IEnumerator(Of ViewRow)`
C#
`public IEnumerator<ViewRow> GetEnumerator()`

Return Value

A IEnumerator that can be used to iterate through the collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

IndexOf Method
The view row to locate in the collection.

Determines the ordinal position of a specific view row in the ViewRowCollection.

Syntax

Visual Basic (Declaration)

```
Public Function IndexOf( _
   ByVal row As ViewRow _
) As Integer
```

C#

```
public int IndexOf(
   ViewRow row
)
```

Parameters

row

The view row to locate in the collection.

Return Value

The ordinal position of the view row in the collection if it is found; otherwise, -1.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

Remove Method
The view row representing the item to delete.

Deletes the specified view item.

Syntax

Visual Basic (Declaration)

```
Public Function Remove( _
   ByVal row As ViewRow _
) As Boolean
```

C#

```
public bool Remove(
   ViewRow row
)
```

Parameters

row

The view row representing the item to delete.

Return Value

true, if the item was deleted as a result of this operation; otherwise, **false**.

Remarks

This is an update operation on the view equivalent to calling ViewRow.Delete.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

RemoveAt Method
The zero-based ordinal position of the item to remove.

Deletes the view row at a specified ordinal position in ViewRowCollection.

Syntax

Visual Basic (Declaration)

```
Public Sub RemoveAt( _
   ByVal ordinal As Integer _
)
```

C#

```
public void RemoveAt(
   int ordinal
)
```

Parameters

ordinal

The zero-based ordinal position of the item to remove.

Remarks

This is an update operation on the view equivalent to calling ViewRow.Delete.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

Properties

For a list of all members of this type, see ViewRowCollection members.

Public Properties

	Name	Description
	AllowClear	Gets a value indicating whether the Clear operation is allowed on the view directly.
	AllowEdit	Gets a value indicating whether modifying property values is allowed on the view directly.
	AllowNew	Gets a value indicating whether the CreateRow operation is allowed on the view directly.
	AllowRemove	Gets a value indicating whether deleting rows is allowed on the view directly.
	Count	Gets the number of elements in the view.
	GroupDescriptions	Gets a collection of System.ComponentModel.GroupDescription objects that describe how the items in the collection are grouped.
	Groups	Gets the top-level groups.
	Item	Gets the view row at the specified ordinal position.
	Properties	Returns the collection of properties available in the view element type to programmatic access through ViewRow and to data binding.
	SortDescriptions	Gets a collection of System.ComponentModel.SortDescription objects that describe how the items in the collection are sorted.

Top

See Also

Reference

ViewRowCollection Class
C1.LiveLinq.LiveViews Namespace

AllowClear Property
Gets a value indicating whether the Clear operation is allowed on the view directly.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property AllowClear As Boolean`
C#
`public bool AllowClear {get;}`

Remarks

As any other update operation performed directly on the view (as opposed to on the base data collection on which that view depends, see C1.LiveLinq.LiveViewExtensions.AsUpdatable<T>), it is allowed only if the view is not read-only (see View.IsReadOnly). Note that the same operation is often allowed on the base data collection, and it will normally result in the corresponding change of the view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

AllowEdit Property
Gets a value indicating whether modifying property values is allowed on the view directly.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property AllowEdit As Boolean`
C#
`public bool AllowEdit {get;}`

Remarks

As any other update operation performed directly on the view (as opposed to on the base data collection on which that view depends, see C1.LiveLinq.LiveViewExtensions.AsUpdatable<T>), it is allowed only if the view is not read-only (see View.IsReadOnly).

Although read-only properties of a view cannot be modified directly in the view, they still reflect up-to-date values of the source, so the difference is often not critical, you can always modify corresponding property in the source, that will automatically change the property in the view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

AllowNew Property
Gets a value indicating whether the CreateRow operation is allowed on the view directly.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property AllowNew As Boolean`
C#

```
public bool AllowNew {get;}
```

Remarks

As any other update operation performed directly on the view (as opposed to on the base data collection on which that view depends, see C1.LiveLinq.LiveViewExtensions.AsUpdatable<T>), it is allowed only if the view is not read-only (see View.IsReadOnly). Note that the same operation is usually allowed on the base data collection, and it will normally result in the corresponding change of the view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

AllowRemove Property
Gets a value indicating whether deleting rows is allowed on the view directly.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Property AllowRemove As Boolean
```

C#

```
public bool AllowRemove {get;}
```

Remarks

As any other update operation performed directly on the view (as opposed to on the base data collection on which that view depends, see C1.LiveLinq.LiveViewExtensions.AsUpdatable<T>), it is allowed only if the view is not read-only (see View.IsReadOnly). Note that the same operation is often allowed on the base data collection, and it will normally result in the corresponding change of the view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

Count Property
Gets the number of elements in the view.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Count As Integer`
C#
`public int Count {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

GroupDescriptions Property
Gets a collection of System.ComponentModel.GroupDescription objects that describe how the items in the collection are grouped.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property GroupDescriptions As ObservableCollection(Of GroupDescription)`
C#
`public ObservableCollection<GroupDescription> GroupDescriptions {get;}`

Property Value

A collection of System.ComponentModel.GroupDescription objects that describe how the items in the collection are grouped.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

Groups Property
Gets the top-level groups.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property Groups As ReadOnlyObservableCollection(Of Object)`
C#
`public ReadOnlyObservableCollection<object> Groups {get;}`

Property Value

A read-only collection of the top-level groups.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

Item Property
The zero-based ordinal position of the view row.

Gets the view row at the specified ordinal position.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Default Property Item( _
   ByVal ordinal As Integer _
) As ViewRow
```

C#

```
public ViewRow this[
   int ordinal
]; {get;}
```

Parameters

ordinal

The zero-based ordinal position of the view row.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

Properties Property
Returns the collection of properties available in the view element type to programmatic access through ViewRow and to data binding.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Property Properties As ViewRowPropertyInfo()
```

C#

```
public ViewRowPropertyInfo[] Properties {get;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

SortDescriptions Property
Gets a collection of System.ComponentModel.SortDescription objects that describe how the items in the collection are sorted.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Property SortDescriptions As SortDescriptionCollection
```

C#

```
public SortDescriptionCollection SortDescriptions {get;}
```

Property Value

A collection of System.ComponentModel.SortDescription objects that describe how the items in the collection are sorted.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

Events
For a list of all members of this type, see ViewRowCollection members.

Public Events

	Name	Description
	Changed	Occurs after a view row has changed.
	ViewRowAdding	Occurs after a new view row is created so it can be populated with default values.

Top

See Also

Reference

ViewRowCollection Class
C1.LiveLinq.LiveViews Namespace

Changed Event
Occurs after a view row has changed.

Syntax

Visual Basic (Declaration)
`Public Event Changed As EventHandler(Of SourceChangeEventArgs(Of ViewRow))`
C#
`public event EventHandler<SourceChangeEventArgs<ViewRow>> Changed`

Event Data

The event handler receives an argument of type SourceChangeEventArgs<T> containing data related to this event. The following **SourceChangeEventArgs<T>** properties provide information specific to this event.

Property	Description
ChangeType	Gets the type of change.
Item	Gets the object that is being changed.
Ordinal	Gets the ordinal position of the collection item that is being changed.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

ViewRowAdding Event
Occurs after a new view row is created so it can be populated with default values.

Syntax

Visual Basic (Declaration)

```
Public Event ViewRowAdding As EventHandler(Of ViewRowAddingEventArgs)
```

C#

```
public event EventHandler<ViewRowAddingEventArgs> ViewRowAdding
```

Event Data

The event handler receives an argument of type ViewRowAddingEventArgs containing data related to this event. The following **ViewRowAddingEventArgs** properties provide information specific to this event.

Property	Description
Row	Gets the new view row that has just been added to ViewRowCollection.

Remarks

This event occurs only if the new row is created directly in the view, as a result of a view update operation, that is, with CreateRow or via data binding. It does not occur when new rows appear in the view as a result of view maintenance on changes made to the view's source data collections.

This event occurs immediately on creating the new row, before the method creating it returns, before the row enters edit mode (see ViewRowState about edit mode).

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowCollection Class
ViewRowCollection Members

ViewRowPropertyInfo
Allows to control certain behavior of a property of the element type of a View.

Object Model

ViewRowPropertyInfo

Syntax

Visual Basic (Declaration)
`Public Class ViewRowPropertyInfo`
C#
`public class ViewRowPropertyInfo`

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.ViewRowPropertyInfo

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowPropertyInfo Members
C1.LiveLinq.LiveViews Namespace

Overview
Allows to control certain behavior of a property of the element type of a View.

Object Model

ViewRowPropertyInfo

Syntax

Visual Basic (Declaration)
`Public Class ViewRowPropertyInfo`

C#
`public class ViewRowPropertyInfo`

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.ViewRowPropertyInfo

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowPropertyInfo Members
C1.LiveLinq.LiveViews Namespace

Members
Properties

The following tables list the members exposed by ViewRowPropertyInfo.

Public Properties

	Name	Description
	ImmediateUpdate	Gets or sets a boolean value indicating whether changes made to this property through data binding must be immediately sent to the corresponding view item even if the view in editing mode.
	PropertyName	Gets the name of the property.

Top

See Also

Reference

ViewRowPropertyInfo Class
C1.LiveLinq.LiveViews Namespace

Properties
>

Name	Description
ImmediateUpdate	Gets or sets a boolean value indicating whether changes made to this property through data binding must be immediately sent to the corresponding view item even if the view in editing mode.
PropertyName	Gets the name of the property.

Top

See Also

Reference

ViewRowPropertyInfo Class
C1.LiveLinq.LiveViews Namespace

ImmediateUpdate Property
Gets or sets a boolean value indicating whether changes made to this property through data binding must be immediately sent to the corresponding view item even if the view in editing mode.

Syntax

Visual Basic (Declaration)

```
Public Property ImmediateUpdate As Boolean
```

C#

```
public bool ImmediateUpdate {get; set;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowPropertyInfo Class
ViewRowPropertyInfo Members
BeginEdit Method

PropertyName Property
Gets the name of the property.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property PropertyName As String`
C#
`public string PropertyName {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ViewRowPropertyInfo Class
ViewRowPropertyInfo Members

Enumerations

ViewMaintenanceMode
Specifies how a view is synchronized with changes in its base data.

Syntax

Visual Basic (Declaration)
`Public Enum ViewMaintenanceMode` `  Inherits System.Enum`

C#	
`public enum ViewMaintenanceMode : System.Enum`	

Members

Member	Description
Default	By default, the view is in a "smart mode": It is in **Deferred** mode if nobody is interested in its data, and it is synchronized and goes to **Immediate** mode if there is a client interested in receiving notifications of changes in that view's data. In other words, a view in **Default** mode is effectively in **Deferred** mode if no listeners registered to be notified of its changes, and it is effectively in **Immediate** mode if there are such listeners (for example, if there is GUI control bound to it).
Deferred	When a change in base data occur, the view is not synchronized with it immediately. It is allowed to go stale, out of sync with its base data as long as there are no requests for this view's data. The view is synchronized on demand, that is, it is synchronized when any request for its data arrives.
Immediate	The view is synchronized with its base data automatically and immediately after any change in its base data occurs. The view is kept synchronized with its base data at all times.

Remarks

See Also: View Maintenance Mode.

Inheritance Hierarchy

System.Object
  System.ValueType
    System.Enum
      C1.LiveLinq.LiveViews.ViewMaintenanceMode

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1.LiveLinq.LiveViews Namespace
MaintenanceMode Property
DeferredMaintenance Property

ViewOrder
Specifies whether and how a view must preserve item order if it exists in the source.

Syntax

Visual Basic (Declaration)

```
Public Enum ViewOrder
   Inherits System.Enum
```

C#

```
public enum ViewOrder : System.Enum
```

Members

Member	Description
NotPreserved	Source order is not preserved. Preserving source order is not guaranteed even at view creation.
PartiallyPreserved	Source order is partially preserved. When the view is created, it preserves source order, but later, when changes occur in the source, view items added or modified to reflect those changes aren't guaranteed to appear at the same order position in the view as in the source.
Preserved	Source order is preserved completely. Order of items in the view is always the same as in the source (if source has a particular order), even after the view is

	maintained to reflect changes that occurred in the source.

Inheritance Hierarchy

System.Object
  System.ValueType
    System.Enum
      C1.LiveLinq.LiveViews.ViewOrder

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1.LiveLinq.LiveViews Namespace

ViewRowState
The state of a view row with regard to edit, add and delete operations if they are performed directly on the view.

Syntax

Visual Basic (Declaration)
`Public Enum ViewRowState` `  Inherits System.Enum`
C#
`public enum ViewRowState : System.Enum`

Members

Member	Description
Detached	The row was deleted, or it is a new row after exiting edit mode.

Modified	The row is in edit mode (BeginEdit was called, neither EndEdit nor ViewRow.CancelEdit was called yet), and the row is not new (was not created by CreateRow).
New	The row was added to the view directly (not by adding to a basic data collection) by calling CreateRow or through data binding (such new row enters edit mode once it is created) and still in edit mode (neither EndEdit nor ViewRow.CancelEdit was called yet).
Unmodified	The row is a regular view row, not in edit mode and not deleted.

Remarks

This state concerns edit, add and delete operations performed directly on the view (programmatically via ViewRow objects or through data binding). It does not concern modifications made to the view's base (source) data collections. Modifications made to source data can also change view items, as a result of the normal view maintenance process, see MaintenanceMode, but in that case the state of thus added or modified rows remains **Unmodified**.

Note on adding rows directly to the view:

If a row is added directly to the view (as opposed to adding it to one of its base data collections), the following happens:

When a new row is created with CreateRow or through data binding, it enters edit mode.

When it is committed with EndEdit, a new row is added to the view's base data collection, and, usually, a corresponding row appears in the view, that has **Unmodified** state, although in some cases it may be more than one row or none, depending on the view query. The original view row no longer corresponds to a view item after the EndEdit or ViewRow.CancelEdit call, its state becomes **Detached**. Accessing it after that would throw an exception.

Inheritance Hierarchy

System.Object
  System.ValueType
    System.Enum
      C1.LiveLinq.LiveViews.ViewRowState

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1.LiveLinq.LiveViews Namespace

C1.LiveLinq.LiveViews.Xml Namespace

Overview

Classes

	Class	Description
	XmlExtensions	Provides a set of static (extension) methods for LiveLinq to XML.

See Also

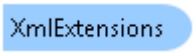
Reference

C1.Silverlight.LiveLinq Assembly

Classes

XmlExtensions
Provides a set of static (extension) methods for LiveLinq to XML.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit NotInheritable Class XmlExtensions`
C#
`public static class XmlExtensions`

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.Xml.XmlExtensions

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

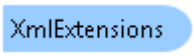
See Also

Reference

XmlExtensions Members
C1.LiveLinq.LiveViews.Xml Namespace

Overview
Provides a set of static (extension) methods for LiveLinq to XML.

Object Model



Syntax

Visual Basic (Declaration)
`Public MustInherit NotInheritable Class XmlExtensions`
C#
`public static class XmlExtensions`

Inheritance Hierarchy

System.Object
  C1.LiveLinq.LiveViews.Xml.XmlExtensions

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Members
C1.LiveLinq.LiveViews.Xml Namespace

Members
Methods

The following tables list the members exposed by XmlExtensions.

Public Methods

	Name	Description
S	AsLive	Overloaded. Creates a view based on the specified XML document.
S	Attributes	Overloaded. Returns a view representing the collection of the attributes of every element in the source view.
S	BeginUpdate	Suspends notifications while massive changes are being made to an XML node and its descendants.
S	DescendantNodes<T>	Returns a view representing the collection of the descendent nodes of every element and document in the source view.
S	DescendantNodesAndSelf	Returns a view representing the collection of nodes that contains every element in the source view, and the descendent nodes of every element in the source view.
S	Descendants	Overloaded. Returns a view representing the collection of elements that contains the descendent elements of every element and document in the source view.

S	DescendantsAndSelf	Overloaded. Returns a view representing a collection of elements that contains every element in the source view, and the descendent elements of every element in the source view.
S	Elements	Overloaded. Returns a view representing the collection of child elements of every element and document in the source view..
S	EndUpdate	Ends notification suspension started with BeginUpdate.
S	Nodes<T>	Returns a view representing the collection of child nodes of every document and element in the source view.
S	Root	Gets the view for the root element of the XML tree for this document.

Top

See Also

Reference

XmlExtensions Class
C1.LiveLinq.LiveViews.Xml Namespace

Methods
>

Name	Description
S AsLive	Overloaded. Creates a view based on the specified XML document.
S Attributes	Overloaded. Returns a view representing the collection of the attributes of every element in the source view.
S BeginUpdate	Suspends notifications while massive changes are being made to an XML node and its descendants.
S DescendantNodes<T>	Returns a view representing the collection of the descendent nodes of every element and document in the source view.

DescendantNodesAndSelf	Returns a view representing the collection of nodes that contains every element in the source view, and the descendent nodes of every element in the source view.
Descendants	Overloaded. Returns a view representing the collection of elements that contains the descendent elements of every element and document in the source view.
DescendantsAndSelf	Overloaded. Returns a view representing a collection of elements that contains every element in the source view, and the descendent elements of every element in the source view.
Elements	Overloaded. Returns a view representing the collection of child elements of every element and document in the source view..
EndUpdate	Ends notification suspension started with BeginUpdate.
Nodes<T>	Returns a view representing the collection of child nodes of every document and element in the source view.
Root	Gets the view for the root element of the XML tree for this document.

Top

See Also

Reference

XmlExtensions Class
C1.LiveLinq.LiveViews.Xml Namespace

AsLive Method
Creates a view based on the specified XML document.

Overload List

Overload	Description
AsLive(XDocument)	Creates a view based on the specified XML document.
AsLive(XElement)	Creates a view based on the specified XML element.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members

AsLive(XDocument) Method
The XML document to expose as a view.

Creates a view based on the specified XML document.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function AsLive( _`
`  ByVal document As XDocument _`
`) As View(Of XDocument)`

C#

`public static View<XDocument> AsLive(`
`  XDocument document`
`)`

Parameters

document

The XML document to expose as a view.

Return Value

A view representing the specified XML document.

Remarks

Since there can be only one root element in a document, the view based on a document (`document.AsLive()`) is similar to the view based on its root element (`document.Root.AsLive()`). The difference is that the document view contains the document as its only item, and the root view contains the root as its only item. This difference is essential only when the root of the document is replaced with a different element (and so the whole XML tree changes), then the document view remains valid and shows the changed document contents, whereas the root view becomes disconnected from the document.

Note: This view is owned by the System.Xml.Linq.XDocument object (it is stored as one of its annotations), so, if you create a view for the same document several times, it will be the same object.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members
Overload List

AsLive(XElement) Method
The XML element to expose as a view.

Creates a view based on the specified XML element.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function AsLive( _
   ByVal element As XElement _
) As View(Of XElement)
```

C#

```
public static View<XElement> AsLive(
   XElement element
```

```
)
```

Parameters

element

The XML element to expose as a view.

Return Value

A view representing the specified XML element.

Remarks

This view represents a single XML node. Therefore, as a collection of items, this view's **Count** is always 1. This view is usually used as a starting point to construct a view containing elements or attributes from this node's descendants by using a query with operators from XmlExtensions such as **Elements**, **Descendants** and others, in combination with standard LINQ query operators **where**, **join** and others.

Note: This view is owned by the System.Xml.Linq.XElement object (it is stored as one of its annotations), so, if you create a view for the same element several times, it will be the same object.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members
Overload List

Attributes Method
Returns a view representing the collection of the attributes of every element in the source view.

Overload List

Overload	Description

Attributes(View<XElement>)	Returns a view representing the collection of the attributes of every element in the source view.
Attributes(View<XElement>,XName)	Returns a view representing a filtered collection of the attributes of every element in the source view. Only attributes that have a matching System.Xml.Linq.XName are included.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members

Attributes(View<XElement>) Method
The source view.

Returns a view representing the collection of the attributes of every element in the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function Attributes( _
   ByVal view As View(Of XElement) _
) As View(Of XAttribute)
```

C#

```
public static View<XAttribute> Attributes(
   View<XElement> view
)
```

Parameters

view

The source view.

Return Value

A view containing the attributes of every element in the source view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members
Overload List

Attributes(View<XElement>,XName) Method
The source view.

The System.Xml.Linq.XName to match.

Returns a view representing a filtered collection of the attributes of every element in the source view. Only attributes that have a matching System.Xml.Linq.XName are included.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function Attributes( _
   ByVal view As View(Of XElement), _
   ByVal name As XName _
) As View(Of XAttribute)
```

C#

```
public static View<XAttribute> Attributes(
   View<XElement> view,
   XName name
)
```

Parameters

view

The source view.

name

The System.Xml.Linq.XName to match.

Return Value

A view that contains a filtered collection of the attributes of every element in the source view. Only attributes that have a matching System.Xml.Linq.XName are included.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members
Overload List

BeginUpdate Method

The node that is the root of a tree where massive changes are made in code.

Suspends notifications while massive changes are being made to an XML node and its descendants.

Syntax

Visual Basic (Declaration)

```
Public Shared Sub BeginUpdate( _
   ByVal node As XContainer _
)
```

C#

```
public static void BeginUpdate(
   XContainer node
)
```

Parameters

node

The node that is the root of a tree where massive changes are made in code.

Remarks

This method must be followed by EndUpdate.

Use this method when you already have indexes over this XML or live views based on it, and you need to perform massive changes on the contents of this node and its descendants. Without this method, every single change you make causes LiveLinq to perform necessary operations for maintaining your indexes and live views dependent on this node and its descendants. In case of massive changes, this can be slower than to wait until the massive changes are done and rebuild the indexes and live views.

Between **BeginUpdate** and EndUpdate calls, indexes, live views, bound controls and other change notification listeners are not updated, they don't receive change notifications. When EndUpdate is called, a SourceChangeType.Reset notification is sent, meaning all indexes, live views and other collections dependent on this node and its descendants must be rebuilt from scratch.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members

DescendantNodes<T> Method

The type of the objects in the source *view*, constrained to System.Xml.Linq.XContainer.

The source view.

Returns a view representing the collection of the descendent nodes of every element and document in the source view.

Syntax

Visual Basic (Declaration)

```
Public Shared Function DescendantNodes(Of T As XContainer)( _
   ByVal view As View(Of T) _
) As View(Of XNode)
```

C#

```
public static View<XNode> DescendantNodes<T>(
   View<T> view
)
where T: XContainer
```

Parameters

view

The source view.

Type Parameters

T

The type of the objects in the source *view*, constrained to System.Xml.Linq.XContainer.

Return Value

A view containing every descendent node of every document and element in the source view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members

DescendantNodesAndSelf Method
The source view.

Returns a view representing the collection of nodes that contains every element in the source view, and the descendent nodes of every element in the source view.

Syntax

Visual Basic (Declaration)

```
Public Shared Function DescendantNodesAndSelf( _
   ByVal view As View(Of XElement) _
) As View(Of XNode)
```

C#

```
public static View<XNode> DescendantNodesAndSelf(
   View<XElement> view
)
```

Parameters

view

The source view.

Return Value

A view containing every element in the source view, and the descendent nodes of every element in the source view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members

Descendants Method
Returns a view representing the collection of elements that contains the descendent elements of every element and document in the source view.

Overload List

Overload	Description
Descendants<T>(View<T>)	Returns a view representing the collection of elements that contains the descendent elements of every element and document in the source view.
Descendants<T>(View<T>,XName)	Returns a view representing a filtered collection of elements that contains the descendent elements of every element and document in the source view. Only elements that have a matching System.Xml.Linq.XName are included.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members

Descendants<T>(View<T>) Method
The type of the objects in the source *view*, constrained to System.Xml.Linq.XContainer.

The source view.

Returns a view representing the collection of elements that contains the descendent elements of every element and document in the source view.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function Descendants(Of T As XContainer)( _
   ByVal view As View(Of T) _
) As View(Of XElement)
```

C#

```
public static View<XElement> Descendants<T>(
   View<T> view
)
where T: XContainer
```

Parameters

view

The source view.

Type Parameters

T

The type of the objects in the source *view*, constrained to System.Xml.Linq.XContainer.

Return Value

A view containing every descendent element of every element and document in the source view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members
Overload List

Descendants<T>(View<T>,XName) Method
The type of the objects in the source *view*, constrained to System.Xml.Linq.XContainer.

The source view.

The System.Xml.Linq.XName to match.

Returns a view representing a filtered collection of elements that contains the descendent elements of every element and document in the source view. Only elements that have a matching System.Xml.Linq.XName are included.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function Descendants(Of T As XContainer)( _
   ByVal view As View(Of T), _
   ByVal name As XName _
) As View(Of XElement)
```

C#

```
public static View<XElement> Descendants<T>(
   View<T> view,
   XName name
)
where T: XContainer
```

Parameters

view

The source view.

name

The System.Xml.Linq.XName to match.

Type Parameters

T

The type of the objects in the source *view*, constrained to System.Xml.Linq.XContainer.

Return Value

A view containing descendants of elements and documents in the source view. Only elements that have a matching System.Xml.Linq.XName are included.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members
Overload List

DescendantsAndSelf Method

Returns a view representing a collection of elements that contains every element in the source view, and the descendent elements of every element in the source view.

Overload List

Overload	Description
DescendantsAndSelf(View<XElement>)	Returns a view representing a collection of elements that contains every element in the source view, and the descendent elements of every element in the source view.
DescendantsAndSelf(View<XElement>,XName)	Returns a view representing a filtered collection of elements that contains every element in the source view, and the descendants of every element in the source view. Only elements that have a matching System.Xml.Linq.XName are included.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members

DescendantsAndSelf(View<XElement>) Method
The source view.

Returns a view representing a collection of elements that contains every element in the source view, and the descendent elements of every element in the source view.

Syntax

Visual Basic (Declaration)

`Public Overloads Shared Function DescendantsAndSelf( _`
`  ByVal view As View(Of XElement) _`
`) As View(Of XElement)`

C#

`public static View<XElement> DescendantsAndSelf(`
`  View<XElement> view`
`)`

Parameters

view

The source view.

Return Value

A view containing every element in the source view, and the descendent elements of every element in the source view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members
Overload List

DescendantsAndSelf(View<XElement>,XName) Method
The source view.

The System.Xml.Linq.XName to match.

Returns a view representing a filtered collection of elements that contains every element in the source view, and the descendants of every element in the source view. Only elements that have a matching System.Xml.Linq.XName are included.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function DescendantsAndSelf( _
   ByVal view As View(Of XElement), _
   ByVal name As XName _
) As View(Of XElement)
```

C#

```
public static View<XElement> DescendantsAndSelf(
   View<XElement> view,
   XName name
)
```

Parameters

view

The source view.

name

The System.Xml.Linq.XName to match.

Return Value

A view containing elements in the source view, and the descendants of elements in the source view. Only elements that have a matching System.Xml.Linq.XName are included.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members
Overload List

Elements Method
Returns a view representing the collection of child elements of every element and document in the source view..

Overload List

Overload	Description
Elements<T>(View<T>)	Returns a view representing the collection of child elements of every element and document in the source view..
Elements<T>(View<T>,XName)	Returns a view representing filtered collection of child elements of every element and document in the source view. Only elements that have a matching System.Xml.Linq.XName are included.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members

Elements<T>(View<T>) Method
The type of the objects in the source *view*, constrained to System.Xml.Linq.XContainer.

The source view.

Returns a view representing the collection of child elements of every element and document in the source view..

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function Elements(Of T As XContainer)( _
   ByVal view As View(Of T) _
) As View(Of XElement)
```

C#

```
public static View<XElement> Elements<T>(
   View<T> view
)
where T: XContainer
```

Parameters

view

The source view.

Type Parameters

T

The type of the objects in the source *view*, constrained to System.Xml.Linq.XContainer.

Return Value

A view containing the child elements of every element and document in the source view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members
Overload List

Elements<T>(View<T>,XName) Method

The type of the objects in the source *view*, constrained to System.Xml.Linq.XContainer.

The source view.

The System.Xml.Linq.XName to match.

Returns a view representing filtered collection of child elements of every element and document in the source view. Only elements that have a matching System.Xml.Linq.XName are included.

Syntax

Visual Basic (Declaration)

```
Public Overloads Shared Function Elements(Of T As XContainer)( _
   ByVal view As View(Of T), _
   ByVal name As XName _
) As View(Of XElement)
```

C#

```
public static View<XElement> Elements<T>(
   View<T> view,
   XName name
)
where T: XContainer
```

Parameters

view

The source view.

name

The System.Xml.Linq.XName to match.

Type Parameters

T

The type of the objects in the source *view*, constrained to System.Xml.Linq.XContainer.

Return Value

A view that contains a filtered collection of child elements of every element and document in the source view. Only elements that have a matching System.Xml.Linq.XName are included.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members
Overload List

EndUpdate Method

The node that is the root of a tree where massive changes have been made since the BeginUpdate call.

Ends notification suspension started with BeginUpdate.

Syntax

Visual Basic (Declaration)

```
Public Shared Sub EndUpdate( _
   ByVal node As XContainer _
)
```

C#

```
public static void EndUpdate(
   XContainer node
)
```

Parameters

node

The node that is the root of a tree where massive changes have been made since the BeginUpdate call.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members

Nodes<T> Method
The type of the objects in the source *view*, constrained to System.Xml.Linq.XContainer.

The source view.

Returns a view representing the collection of child nodes of every document and element in the source view.

Syntax

Visual Basic (Declaration)

```
Public Shared Function Nodes(Of T As XContainer)( _
   ByVal view As View(Of T) _
) As View(Of XNode)
```

C#

```
public static View<XNode> Nodes<T>(
   View<T> view
)
where T: XContainer
```

Parameters

view

The source view.

Type Parameters

T

The type of the objects in the source *view*, constrained to System.Xml.Linq.XContainer.

Return Value

A view containing every child node of every document and element in the source view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members

Root Method
The view representing this XML document.

Gets the view for the root element of the XML tree for this document.

Syntax

Visual Basic (Declaration)

`Public Shared Function Root( _`
`  ByVal documentView As View(Of XDocument) _`
`) As View(Of XElement)`

C#

`public static View<XElement> Root(`
`  View<XDocument> documentView`
`)`

Parameters

documentView

The view representing this XML document.

Return Value

A view containing a single element, the root of the XML tree.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

XmlExtensions Class
XmlExtensions Members

C1.LiveLinq.Metadata Namespace

Overview
Inheritance Hierarchy

See Also

Reference

C1.Silverlight.LiveLinq Assembly

C1.Win.Data.Entity.4 Assembly

Overview

%%description%%

" -->

Namespaces

Namespace	Description
C1.Win.Data	
C1.Win.Data.Entities	

Namespaces

C1.Win.Data Namespace

Overview

Classes

	Class	Description
	ControlHandler	Represents a control handler that connect GUI controls of supported types to a C1DataSource so that those controls can be given additional functionality such as lookup columns and virtual mode.

See Also

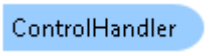
Reference

C1.Win.Data.Entity.4 Assembly

Classes

ControlHandler
Represents a control handler that connect GUI controls of supported types to a C1DataSource so that those controls can be given additional functionality such as lookup columns and virtual mode.

Object Model



Syntax

Visual Basic (Declaration)

```
Public Class ControlHandler
   Inherits C1.Data.DataSource.BaseControlHandler
```

C#

```
public class ControlHandler : C1.Data.DataSource.BaseControlHandler
```

Remarks

To connect a control handler to a control, put a C1.Win.Data.Entities.C1DataSource on your form and configure the ControlHandler extender property (that appears in your GUI controls when you add a C1.Win.Data.Entities.C1DataSource to the form). Alternatively, use the C1.Win.Data.Entities.C1DataSource.SetControlHandler method in code.

The supported GUI controls are: System.Windows.Forms.DataGridView and C1.Win.C1FlexGrid.C1FlexGrid.

Inheritance Hierarchy

System.Object
  System.Windows.Threading.DispatcherObject
    System.Windows.DependencyObject
      C1.Data.DataSource.BaseControlHandler
        C1.Win.Data.ControlHandler

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

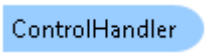
See Also

Reference

ControlHandler Members
C1.Win.Data Namespace

Overview
Represents a control handler that connect GUI controls of supported types to a C1DataSource so that those controls can be given additional functionality such as lookup columns and virtual mode.

Object Model



Syntax

Visual Basic (Declaration)	
`Public Class ControlHandler` `  Inherits C1.Data.DataSource.BaseControlHandler`	

C#

```
public class ControlHandler : C1.Data.DataSource.BaseControlHandler
```

Remarks

To connect a control handler to a control, put a C1.Win.Data.Entities.C1DataSource on your form and configure the ControlHandler extender property (that appears in your GUI controls when you add a C1.Win.Data.Entities.C1DataSource to the form). Alternatively, use the C1.Win.Data.Entities.C1DataSource.SetControlHandler method in code.

The supported GUI controls are: System.Windows.Forms.DataGridView and C1.Win.C1FlexGrid.C1FlexGrid.

Inheritance Hierarchy

System.Object
  System.Windows.Threading.DispatcherObject
    System.Windows.DependencyObject
      C1.Data.DataSource.BaseControlHandler
        C1.Win.Data.ControlHandler

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ControlHandler Members
C1.Win.Data Namespace

Members
Properties Methods

The following tables list the members exposed by ControlHandler.

Public Constructors

	Name	Description

	ControlHandler Constructor	

Top

Public Properties

	Name	Description
	AutoLookup	Gets or sets a value indicating whether data grid columns bound to navigation (foreign key, lookup) properties must be converted to combo box columns, so the user can see the right value and edit it by choosing a value from a drop-down list. The default value is False. (Inherited from C1.Data.DataSource.BaseControlHandler)
	DependencyObjectType	(Inherited from System.Windows.DependencyObject)
	Dispatcher	(Inherited from System.Windows.Threading.DispatcherObject)
	IsSealed	(Inherited from System.Windows.DependencyObject)
	SupportsVirtualMode	Gets a value indicating whether this control handler supports Virtual Mode. (Inherited from C1.Data.DataSource.BaseControlHandler)
	VirtualMode	Gets or sets a value indicating whether virtual mode specified in a C1.Data.DataSource.ClientViewSource is managed by this control handler. (Inherited from C1.Data.DataSource.BaseControlHandler)

Top

Public Methods

	Name	Description
	Apply	Forces this control handler to apply its settings to the current control. (Inherited from C1.Data.DataSource.BaseControlHandler)

	ClearValue	Overloaded. (Inherited from System.Windows.DependencyObject)
	CoerceValue	(Inherited from System.Windows.DependencyObject)
	Equals	(Inherited from System.Windows.DependencyObject)
	GetHashCode	(Inherited from System.Windows.DependencyObject)
	GetLocalValueEnumerator	(Inherited from System.Windows.DependencyObject)
	GetValue	(Inherited from System.Windows.DependencyObject)
	InvalidateProperty	(Inherited from System.Windows.DependencyObject)
	ReadLocalValue	(Inherited from System.Windows.DependencyObject)
	SetCurrentValue	(Inherited from System.Windows.DependencyObject)
	SetValue	Overloaded. (Inherited from System.Windows.DependencyObject)

Top

See Also

Reference

ControlHandler Class
C1.Win.Data Namespace

ControlHandler Constructor

Syntax

Visual Basic (Declaration)
`Public Function New()`
C#
`public ControlHandler()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ControlHandler Class
ControlHandler Members

C1.Win.Data.Entities Namespace

Overview

Classes

	Class	Description
	C1DataSource	A data source control that simplifies data binding of GUI controls to data in a C1.Data.Entities.EntityClientCache. Can be used to bind multiple controls to different queries.
	EntityViewSourceCollection	An observable collection of C1.Data.Entities.EntityViewSource objects.

See Also

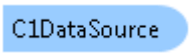
Reference

C1.Win.Data.Entity.4 Assembly

Classes

C1DataSource
A data source control that simplifies data binding of GUI controls to data in a C1.Data.Entities.EntityClientCache. Can be used to bind multiple controls to different queries.

Object Model



Syntax

Visual Basic (Declaration)	
`Public Class C1DataSource` `  Inherits System.ComponentModel.Component`	
C#	
`public class C1DataSource : System.ComponentModel.Component`	

Remarks

To bind a control to data in an C1.Data.Entities.EntityClientCache, put a C1DataSource on your form, specify the context type, populate the ViewSources collection with C1.Data.Entities.EntityViewSource objects to define views (based on queries), and bind the GUI control to the C1DataSource by setting the GUI control's DataSource and DataMember properties.

Inheritance Hierarchy

System.Object
  System.MarshalByRefObject
    System.ComponentModel.Component
      C1.Win.Data.Entities.C1DataSource

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Members
C1.Win.Data.Entities Namespace

Overview
A data source control that simplifies data binding of GUI controls to data in a C1.Data.Entities.EntityClientCache. Can be used to bind multiple controls to different queries.

Object Model

C1DataSource

Syntax

Visual Basic (Declaration)
`Public Class C1DataSource` `  Inherits System.ComponentModel.Component`
C#
`public class C1DataSource : System.ComponentModel.Component`

Remarks

To bind a control to data in an C1.Data.Entities.EntityClientCache, put a C1DataSource on your form, specify the context type, populate the ViewSources collection with C1.Data.Entities.EntityViewSource objects to define views (based on queries), and bind the GUI control to the C1DataSource by setting the GUI control's DataSource and DataMember properties.

Inheritance Hierarchy

System.Object
  System.MarshalByRefObject
    System.ComponentModel.Component
      C1.Win.Data.Entities.C1DataSource

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Members
C1.Win.Data.Entities Namespace

Members
Properties Methods Events

The following tables list the members exposed by C1DataSource.

Public Constructors

	Name	Description
	C1DataSource Constructor	Initializes a new instance of the C1DataSource class.

Top

Public Properties

	Name	Description
	ClientCache	Gets or sets the C1.Data.Entities.EntityClientCache used by this C1DataSource to access the data.
	ClientScope	Gets the client scope to which this C1DataSource belongs.
	Container	(Inherited from System.ComponentModel.Component)
	Item	Overloaded. Gets the C1.Data.DataSource.ClientCollectionView of the C1.Data.Entities.EntityViewSource with the specified name in the ViewSources collection.
	ObjectContext	Gets the System.Data.Objects.ObjectContext the ClientCache is connected to.
	ObjectContextType	Gets or sets the type of an System.Data.Objects.ObjectContext used to obtain the default client cache.
	RefreshInterval	Gets or sets the interval between automatic Refresh operations to refresh the data with any changes that may have occurred on the server.
	ViewSources	Gets a collection of C1.Data.Entities.EntityViewSource objects that define views (based on queries) in this C1DataSource.

Top

Public Methods

	Name	Description
	CreateObjRef	(Inherited from System.MarshalByRefObject)
	Dispose	(Inherited from System.ComponentModel.Component)
	GetControlHandler	Gets the control handler the specified control is currently handled by.
	GetLifetimeService	(Inherited from System.MarshalByRefObject)
	InitializeLifetimeService	(Inherited from System.MarshalByRefObject)
	Load	Loads all C1.Data.Entities.EntityViewSource objects in the ViewSources collection.
	Refresh	Refreshes all C1.Data.Entities.EntityViewSource objects in the ViewSources collection.
	RejectChanges	Rejects the changes for every entity in the ObjectContext.
	SaveChanges	Persists all changes to the server.
	SetControlHandler	Connects a controlHandler to a given control.
	ToString	(Inherited from System.ComponentModel.Component)

Top

Public Events

	Name	Description
	Disposed	(Inherited from System.ComponentModel.Component)

	SavedChanges	Occurs after a save operation is completed.
	SavingChanges	Occurs before changes are saved.

Top

See Also

Reference

C1DataSource Class
C1.Win.Data.Entities Namespace

C1DataSource Constructor
Initializes a new instance of the C1DataSource class.

Syntax

Visual Basic (Declaration)
`Public Function New()`
C#
`public C1DataSource()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

Methods
For a list of all members of this type, see C1DataSource members.

Public Methods

	Name	Description
	CreateObjRef	(Inherited from System.MarshalByRefObject)
	Dispose	(Inherited from System.ComponentModel.Component)
	GetControlHandler	Gets the control handler the specified control is currently handled by.
	GetLifetimeService	(Inherited from System.MarshalByRefObject)
	InitializeLifetimeService	(Inherited from System.MarshalByRefObject)
	Load	Loads all C1.Data.Entities.EntityViewSource objects in the ViewSources collection.
	Refresh	Refreshes all C1.Data.Entities.EntityViewSource objects in the ViewSources collection.
	RejectChanges	Rejects the changes for every entity in the ObjectContext.
	SaveChanges	Persists all changes to the server.
	SetControlHandler	Connects a controlHandler to a given control.
	ToString	(Inherited from System.ComponentModel.Component)

Top

See Also

Reference

C1DataSource Class
C1.Win.Data.Entities Namespace

GetControlHandler Method
The control to get the control handler for.

Gets the control handler the specified control is currently handled by.

Syntax

Visual Basic (Declaration)

```
Public Function GetControlHandler( _
   ByVal control As Control _
) As ControlHandler
```

C#

```
public ControlHandler GetControlHandler(
   Control control
)
```

Parameters

control

The control to get the control handler for.

Return Value

The C1.Win.Data.ControlHandler that currently handles the specified control.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

Load Method
Loads all C1.Data.Entities.EntityViewSource objects in the ViewSources collection.

Syntax

Visual Basic (Declaration)

```
Public Sub Load()
```

C#

```
public void Load()
```

Remarks

This method calls Load for all elements of the ViewSources collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

Refresh Method
Refreshes all C1.Data.Entities.EntityViewSource objects in the ViewSources collection.

Syntax

Visual Basic (Declaration)

```
Public Sub Refresh()
```

C#

```
public void Refresh()
```

Remarks

This method calls C1.Data.DataSource.ClientViewSource.Refresh for all elements of the ViewSources collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

RejectChanges Method
Rejects the changes for every entity in the ObjectContext.

Syntax

Visual Basic (Declaration)
`Public Sub RejectChanges()`
C#
`public void RejectChanges()`

Remarks

Changes will be rejected for all entities in the ObjectContext, including those that were not loaded through this C1DataSource. This will also cancel a pending Add or Edit in the collection views.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

SaveChanges Method
Persists all changes to the server.

Syntax

Visual Basic (Declaration)

```
Public Sub SaveChanges()
```

C#

```
public void SaveChanges()
```

Remarks

Changes will be saved for all entities in the ObjectContext, including those that were not loaded through this C1DataSource. This will also commit a pending Add or Edit in the collection views.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

SetControlHandler Method
The control to be handled by the specified controlHandler.

The control handler to handle the specified control, or null to disconnect the control from the previously connected control handler.

Connects a controlHandler to a given control.

Syntax

Visual Basic (Declaration)

```
Public Sub SetControlHandler( _
   ByVal control As Control, _
   ByVal controlHandler As ControlHandler _
```

```
)
```

C#

```
public void SetControlHandler(
   Control control,
   ControlHandler controlHandler
)
```

Parameters

control

The control to be handled by the specified controlHandler.

controlHandler

The control handler to handle the specified control, or null to disconnect the control from the previously connected control handler.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

Properties
For a list of all members of this type, see C1DataSource members.

Public Properties

	Name	Description
	ClientCache	Gets or sets the C1.Data.Entities.EntityClientCache used by this C1DataSource to access the data.

	ClientScope	Gets the client scope to which this C1DataSource belongs.
	Container	(Inherited from System.ComponentModel.Component)
	Item	Overloaded. Gets the C1.Data.DataSource.ClientCollectionView of the C1.Data.Entities.EntityViewSource with the specified name in the ViewSources collection.
	ObjectContext	Gets the System.Data.Objects.ObjectContext the ClientCache is connected to.
	ObjectContextType	Gets or sets the type of an System.Data.Objects.ObjectContext used to obtain the default client cache.
	RefreshInterval	Gets or sets the interval between automatic Refresh operations to refresh the data with any changes that may have occurred on the server.
	ViewSources	Gets a collection of C1.Data.Entities.EntityViewSource objects that define views (based on queries) in this C1DataSource.

Top

See Also

Reference

C1DataSource Class
C1.Win.Data.Entities Namespace

ClientCache Property
Gets or sets the C1.Data.Entities.EntityClientCache used by this C1DataSource to access the data.

Syntax

Visual Basic (Declaration)
`Public Property ClientCache As EntityClientCache`
C#

```
public EntityClientCache ClientCache {get; set;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

ClientScope Property
Gets the client scope to which this C1DataSource belongs.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property ClientScope As EntityClientScope`
C#
`public EntityClientScope ClientScope {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

Item Property
Gets the C1.Data.DataSource.ClientCollectionView of the C1.Data.Entities.EntityViewSource with the specified name in the ViewSources collection.

Overload List

Overload	Description
Item(String)	Gets the C1.Data.DataSource.ClientCollectionView of the C1.Data.Entities.EntityViewSource with the specified name in the ViewSources collection.
Item(Int32)	Gets the C1.Data.DataSource.ClientCollectionView of the C1.Data.Entities.EntityViewSource at the specified index in the ViewSources collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

Item(String) Property
The name of the C1.Data.Entities.EntityViewSource to take the C1.Data.DataSource.ClientCollectionView from.

Gets the C1.Data.DataSource.ClientCollectionView of the C1.Data.Entities.EntityViewSource with the specified name in the ViewSources collection.

Syntax

Visual Basic (Declaration)

```
Public Overloads ReadOnly Property Item( _
   ByVal name As String _
) As ClientCollectionView
```

C#

```
public ClientCollectionView Item(
   string name
) {get;}
```

Parameters

name

The name of the C1.Data.Entities.EntityViewSource to take the C1.Data.DataSource.ClientCollectionView from.

Property Value

The C1.Data.DataSource.ClientCollectionView of the C1.Data.Entities.EntityViewSource.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members
Overload List

Item(Int32) Property
The index of the C1.Data.Entities.EntityViewSource to take the C1.Data.DataSource.ClientCollectionView from.

Gets the C1.Data.DataSource.ClientCollectionView of the C1.Data.Entities.EntityViewSource at the specified index in the ViewSources collection.

Syntax

Visual Basic (Declaration)

```
Public Overloads ReadOnly Property Item( _
   ByVal index As Integer _
```

```
) As ClientCollectionView
```

C#

```
public ClientCollectionView Item(
   int index
) {get;}
```

Parameters

index

The index of the C1.Data.Entities.EntityViewSource to take the C1.Data.DataSource.ClientCollectionView from.

Property Value

The C1.Data.DataSource.ClientCollectionView of the C1.Data.Entities.EntityViewSource.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members
Overload List

ObjectContext Property

Gets the System.Data.Objects.ObjectContext the ClientCache is connected to.

Syntax

Visual Basic (Declaration)

```
Public ReadOnly Property ObjectContext As ObjectContext
```

C#

```
public ObjectContext ObjectContext {get;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

ObjectContextType Property
Gets or sets the type of an System.Data.Objects.ObjectContext used to obtain the default client cache.

Syntax

Visual Basic (Declaration)
`Public Property ObjectContextType As Type`
C#
`public Type ObjectContextType {get; set;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

RefreshInterval Property
Gets or sets the interval between automatic Refresh operations to refresh the data with any changes that may have occurred on the server.

Syntax

Visual Basic (Declaration)
`Public Property RefreshInterval As TimeSpan`
C#
`public TimeSpan RefreshInterval {get; set;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

ViewSources Property
Gets a collection of C1.Data.Entities.EntityViewSource objects that define views (based on queries) in this C1DataSource.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property ViewSources As EntityViewSourceCollection`
C#
`public EntityViewSourceCollection ViewSources {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

Events
For a list of all members of this type, see C1DataSource members.

Public Events

	Name	Description
	Disposed	(Inherited from System.ComponentModel.Component)
	SavedChanges	Occurs after a save operation is completed.
	SavingChanges	Occurs before changes are saved.

Top

See Also

Reference

C1DataSource Class
C1.Win.Data.Entities Namespace

SavedChanges Event
Occurs after a save operation is completed.

Syntax

Visual Basic (Declaration)
`Public Event SavedChanges As EventHandler(Of SavedChangesEventArgs)`
C#
`public event EventHandler<SavedChangesEventArgs> SavedChanges`

Event Data

The event handler receives an argument of type SavedChangesEventArgs containing data related to this event. The following **SavedChangesEventArgs** properties provide information specific to this event.

Property	Description
Error	Gets a value showing the error that occurred during a save operation.
HasError	Gets a value indicating whether the save operation has failed. If true, inspect the Error property for details.
IsErrorHandled	Gets a value indicating whether the error has been marked as handled by calling MarkErrorAsHandled.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members
SaveChanges Method

SavingChanges Event
Occurs before changes are saved.

Syntax

Visual Basic (Declaration)
`Public Event SavingChanges As EventHandler(Of CancelEventArgs)`
C#
`public event EventHandler<CancelEventArgs> SavingChanges`

Event Data

The event handler receives an argument of type CancelEventArgs containing data related to this event. The following **CancelEventArgs** properties provide information specific to this event.

Property	Description
Cancel	

Remarks

This event is raised from the SaveChanges method and allows a handler to cancel the operation before it begins. When a handler sets System.ComponentModel.CancelEventArgs.Cancel to True, the operation will be aborted and a subsequent SavedChanges event will not be raised.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

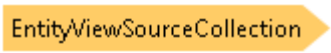
See Also

Reference

C1DataSource Class
C1DataSource Members
SaveChanges Method

EntityViewSourceCollection
An observable collection of C1.Data.Entities.EntityViewSource objects.

Object Model



Syntax

Visual Basic (Declaration)	

```
Public Class EntityViewSourceCollection
   Inherits System.Collections.ObjectModel.ObservableCollection(Of
EntityViewSource)
```

C#

```
public class EntityViewSourceCollection :
System.Collections.ObjectModel.ObservableCollection<EntityViewSource>
```

Inheritance Hierarchy

System.Object
  System.Collections.ObjectModel.Collection<T>
    System.Collections.ObjectModel.ObservableCollection<T>
      C1.Win.Data.Entities.EntityViewSourceCollection

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

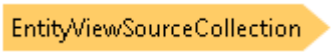
See Also

Reference

EntityViewSourceCollection Members
C1.Win.Data.Entities Namespace

Overview
An observable collection of C1.Data.Entities.EntityViewSource objects.

Object Model



Syntax

Visual Basic (Declaration)

```
Public Class EntityViewSourceCollection
   Inherits System.Collections.ObjectModel.ObservableCollection(Of
EntityViewSource)
```

C#

```
public class EntityViewSourceCollection :
System.Collections.ObjectModel.ObservableCollection<EntityViewSource>
```

Inheritance Hierarchy

System.Object
  System.Collections.ObjectModel.Collection<T>
    System.Collections.ObjectModel.ObservableCollection<T>
      C1.Win.Data.Entities.EntityViewSourceCollection

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

EntityViewSourceCollection Members
C1.Win.Data.Entities Namespace

Members
Properties Methods Events

The following tables list the members exposed by EntityViewSourceCollection.

Public Constructors

	Name	Description
	EntityViewSourceCollection Constructor	

Top

Public Properties

	Name	Description
	Count	(Inherited from System.Collections.ObjectModel.Collection<EntityViewSource>)

	Item	Gets the C1.Data.Entities.EntityViewSource with the given name.

Top

Public Methods

	Name	Description
	Add	(Inherited from System.Collections.ObjectModel.Collection<EntityViewSource>)
	Clear	(Inherited from System.Collections.ObjectModel.Collection<EntityViewSource>)
	Contains	(Inherited from System.Collections.ObjectModel.Collection<EntityViewSource>)
	CopyTo	(Inherited from System.Collections.ObjectModel.Collection<EntityViewSource>)
	GetEnumerator	(Inherited from System.Collections.ObjectModel.Collection<EntityViewSource>)
	IndexOf	(Inherited from System.Collections.ObjectModel.Collection<EntityViewSource>)
	Insert	(Inherited from System.Collections.ObjectModel.Collection<EntityViewSource>)
	Move	(Inherited from System.Collections.ObjectModel.ObservableCollection<EntityViewSource>)
	Remove	(Inherited from System.Collections.ObjectModel.Collection<EntityViewSource>)
	RemoveAt	(Inherited from

		System.Collections.ObjectModel.Collection<EntityViewSource>)

Top

Public Events

	Name	Description
	CollectionChanged	(Inherited from System.Collections.ObjectModel.ObservableCollection<EntityViewSource>)

Top

See Also

Reference

EntityViewSourceCollection Class
C1.Win.Data.Entities Namespace

EntityViewSourceCollection Constructor

Syntax

Visual Basic (Declaration)
`Public Function New()`
C#
`public EntityViewSourceCollection()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

EntityViewSourceCollection Class
EntityViewSourceCollection Members

Properties
For a list of all members of this type, see EntityViewSourceCollection members.

Public Properties

	Name	Description
	Count	(Inherited from System.Collections.ObjectModel.Collection<EntityViewSource>)
	Item	Gets the C1.Data.Entities.EntityViewSource with the given name.

Top

See Also

Reference

EntityViewSourceCollection Class
C1.Win.Data.Entities Namespace

Item Property
The name of the C1.Data.Entities.EntityViewSource to get from the collection.

Gets the C1.Data.Entities.EntityViewSource with the given name.

Syntax

Visual Basic (Declaration)

```
Public Shadows ReadOnly Default Property Item( _
   ByVal name As String _
) As EntityViewSource
```

C#

```
public new EntityViewSource this[
   string name
]; {get;}
```

Parameters

name

The name of the C1.Data.Entities.EntityViewSource to get from the collection.

Property Value

The C1.Data.Entities.EntityViewSource with the specified name, or null if it does not exist.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

EntityViewSourceCollection Class
EntityViewSourceCollection Members

C1.WPF.Data.Entity.4 Assembly

Overview

%%description%%

" -->

Namespaces

Namespace	Description
C1.WPF.Data	
C1.WPF.Data.Entities	

Namespaces

C1.WPF.Data Namespace

Overview

Classes

	Class	Description

	ControlHandler	Represents a control handler that connect GUI controls of supported types to a C1DataSource so that those controls can be given additional functionality such as lookup columns and virtual mode.

See Also

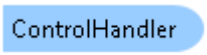
Reference

C1.WPF.Data.Entity.4 Assembly

Classes

ControlHandler
Represents a control handler that connect GUI controls of supported types to a C1DataSource so that those controls can be given additional functionality such as lookup columns and virtual mode.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class ControlHandler` `  Inherits C1.Data.DataSource.BaseControlHandler`
C#
`public class ControlHandler : C1.Data.DataSource.BaseControlHandler`

Remarks

To connect a control handler to a GUI control, assign an instance of this class to the C1.WPF.Data.Entities.C1DataSource.ControlHandlerProperty attached property of the GUI control.

The supported GUI controls are: System.Windows.Controls.DataGrid, C1.WPF.FlexGrid.C1FlexGrid, C1.WPF.DataGrid.C1DataGrid.

Inheritance Hierarchy

System.Object
  System.Windows.Threading.DispatcherObject

System.Windows.DependencyObject
C1.Data.DataSource.BaseControlHandler
C1.WPF.Data.ControlHandler

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

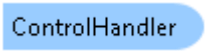
See Also

Reference

ControlHandler Members
C1.WPF.Data Namespace

Overview
Represents a control handler that connect GUI controls of supported types to a C1DataSource so that those controls can be given additional functionality such as lookup columns and virtual mode.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class ControlHandler` `  Inherits C1.Data.DataSource.BaseControlHandler`
C#
`public class ControlHandler : C1.Data.DataSource.BaseControlHandler`

Remarks

To connect a control handler to a GUI control, assign an instance of this class to the C1.WPF.Data.Entities.C1DataSource.ControlHandlerProperty attached property of the GUI control.

The supported GUI controls are: System.Windows.Controls.DataGrid, C1.WPF.FlexGrid.C1FlexGrid, C1.WPF.DataGrid.C1DataGrid.

Inheritance Hierarchy

System.Object
  System.Windows.Threading.DispatcherObject
    System.Windows.DependencyObject
      C1.Data.DataSource.BaseControlHandler
        C1.WPF.Data.ControlHandler

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ControlHandler Members
C1.WPF.Data Namespace

Members
Fields Properties Methods

The following tables list the members exposed by ControlHandler.

Public Constructors

	Name	Description
	ControlHandler Constructor	

Top

Public Fields

	Name	Description
S	DataSourceProperty	The DependencyProperty for the DataSource property.

Top

Public Properties

	Name	Description
	AutoLookup	Gets or sets a value indicating whether data grid columns bound to navigation (foreign key, lookup) properties must be converted to combo box columns, so the user can see the right value and edit it by choosing a value from a drop-down list. The default value is False. (Inherited from C1.Data.DataSource.BaseControlHandler)
	DataSource	Gets or sets a data source of the control handler.
	DependencyObjectType	(Inherited from System.Windows.DependencyObject)
	Dispatcher	(Inherited from System.Windows.Threading.DispatcherObject)
	IsSealed	(Inherited from System.Windows.DependencyObject)
	SupportsVirtualMode	Gets a value indicating whether this control handler supports Virtual Mode. (Inherited from C1.Data.DataSource.BaseControlHandler)
	VirtualMode	Gets or sets a value indicating whether virtual mode specified in a C1.Data.DataSource.ClientViewSource is managed by this control handler. (Inherited from C1.Data.DataSource.BaseControlHandler)

Top

Public Methods

	Name	Description
	Apply	Forces this control handler to apply its settings to the current control. (Inherited from C1.Data.DataSource.BaseControlHandler)
	ClearValue	Overloaded. (Inherited from System.Windows.DependencyObject)
	CoerceValue	(Inherited from System.Windows.DependencyObject)

	Equals	(Inherited from System.Windows.DependencyObject)
	GetHashCode	(Inherited from System.Windows.DependencyObject)
	GetLocalValueEnumerator	(Inherited from System.Windows.DependencyObject)
	GetValue	(Inherited from System.Windows.DependencyObject)
	InvalidateProperty	(Inherited from System.Windows.DependencyObject)
	ReadLocalValue	(Inherited from System.Windows.DependencyObject)
	SetCurrentValue	(Inherited from System.Windows.DependencyObject)
	SetValue	Overloaded. (Inherited from System.Windows.DependencyObject)

Top

See Also

Reference

ControlHandler Class
C1.WPF.Data Namespace

ControlHandler Constructor

Syntax

Visual Basic (Declaration)
`Public Function New()`
C#
`public ControlHandler()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ControlHandler Class
ControlHandler Members

Properties
For a list of all members of this type, see ControlHandler members.

Public Properties

	Name	Description
	AutoLookup	Gets or sets a value indicating whether data grid columns bound to navigation (foreign key, lookup) properties must be converted to combo box columns, so the user can see the right value and edit it by choosing a value from a drop-down list. The default value is False. (Inherited from C1.Data.DataSource.BaseControlHandler)
	DataSource	Gets or sets a data source of the control handler.
	DependencyObjectType	(Inherited from System.Windows.DependencyObject)
	Dispatcher	(Inherited from System.Windows.Threading.DispatcherObject)
	IsSealed	(Inherited from System.Windows.DependencyObject)
	SupportsVirtualMode	Gets a value indicating whether this control handler supports Virtual Mode. (Inherited from C1.Data.DataSource.BaseControlHandler)
	VirtualMode	Gets or sets a value indicating whether virtual mode specified in a C1.Data.DataSource.ClientViewSource is managed by this control handler. (Inherited from C1.Data.DataSource.BaseControlHandler)

Top

See Also

Reference

ControlHandler Class
C1.WPF.Data Namespace

DataSource Property
Gets or sets a data source of the control handler.

Syntax

Visual Basic (Declaration)
`Public Property DataSource As C1DataSource`
C#
`public C1DataSource DataSource {get; set;}`

Remarks

Setting this property is required if the GUI control is not bound directly to a C1DataSource. For example, it must be set if the GUI control is bound to a live view.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ControlHandler Class
ControlHandler Members

Fields
For a list of all members of this type, see ControlHandler members.

Public Fields

	Name	Description

S	DataSourceProperty	The DependencyProperty for the DataSource property.

Top

See Also

Reference

ControlHandler Class
C1.WPF.Data Namespace

DataSourceProperty Field
The DependencyProperty for the DataSource property.

Syntax

Visual Basic (Declaration)
`Public Shared ReadOnly DataSourceProperty As DependencyProperty`
C#
`public static readonly DependencyProperty DataSourceProperty`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

ControlHandler Class
ControlHandler Members

C1.WPF.Data.Entities Namespace

Overview

Classes

	Class	Description

	C1DataSource	A data source control that simplifies data binding of GUI controls to data in a C1.Data.Entities.EntityClientCache. Can be used to bind multiple controls to different queries.
	EntityViewSourceCollection	An observable collection of C1.Data.Entities.EntityViewSource objects.

See Also

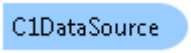
Reference

C1.WPF.Data.Entity.4 Assembly

Classes

C1DataSource
A data source control that simplifies data binding of GUI controls to data in a C1.Data.Entities.EntityClientCache. Can be used to bind multiple controls to different queries.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class C1DataSource` `  Inherits System.Windows.Controls.Control`
C#
`public class C1DataSource : System.Windows.Controls.Control`

Remarks

To bind a control to data in a C1.Data.Entities.EntityClientCache, add a C1DataSource to a XAML file, specify the context type, populate the ViewSources collection with C1.Data.Entities.EntityViewSource objects to define views (based on queries), and bind GUI controls like this: `<DataGrid ItemsSource="{Binding Customers, ElementName=c1DataSource}"/>` where Customers is the name of an

C1.Data.Entities.EntityViewSource in the ViewSources collection and c1DataSource is the name of the C1DataSource.

Inheritance Hierarchy

System.Object
  System.Windows.Threading.DispatcherObject
    System.Windows.DependencyObject
      System.Windows.Media.Visual
        System.Windows.UIElement
          System.Windows.FrameworkElement
            System.Windows.Controls.Control
              C1.WPF.Data.Entities.C1DataSource

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

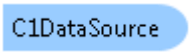
See Also

Reference

C1DataSource Members
C1.WPF.Data.Entities Namespace

Overview
A data source control that simplifies data binding of GUI controls to data in a C1.Data.Entities.EntityClientCache. Can be used to bind multiple controls to different queries.

Object Model



Syntax

Visual Basic (Declaration)	
`Public Class C1DataSource` `  Inherits System.Windows.Controls.Control`	

C#

```
public class C1DataSource : System.Windows.Controls.Control
```

Remarks

To bind a control to data in a C1.Data.Entities.EntityClientCache, add a C1DataSource to a XAML file, specify the context type, populate the ViewSources collection with C1.Data.Entities.EntityViewSource objects to define views (based on queries), and bind GUI controls like this: `<DataGrid ItemsSource="{Binding Customers, ElementName=c1DataSource}"/>` where Customers is the name of an C1.Data.Entities.EntityViewSource in the ViewSources collection and c1DataSource is the name of the C1DataSource.

Inheritance Hierarchy

System.Object
  System.Windows.Threading.DispatcherObject
    System.Windows.DependencyObject
      System.Windows.Media.Visual
        System.Windows.UIElement
          System.Windows.FrameworkElement
            System.Windows.Controls.Control
              C1.WPF.Data.Entities.C1DataSource

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Members
C1.WPF.Data.Entities Namespace

Members
Fields Properties Methods Events

The following tables list the members exposed by C1DataSource.

Public Constructors

	Name	Description
	C1DataSource Constructor	Initializes a new instance of the C1DataSource class.

Top

Public Fields

	Name	Description
S	ControlHandlerProperty	Identifies the C1.WPF.Data.Entities.C1DataSource.ControlHandler attached property.

Top

Public Properties

	Name	Description
	ActualHeight	(Inherited from System.Windows.FrameworkElement)
	ActualWidth	(Inherited from System.Windows.FrameworkElement)
	AllowDrop	(Inherited from System.Windows.UIElement)
	AreAnyTouchesCaptured	(Inherited from System.Windows.UIElement)
	AreAnyTouchesCapturedWithin	(Inherited from System.Windows.UIElement)
	AreAnyTouchesDirectlyOver	(Inherited from System.Windows.UIElement)
	AreAnyTouchesOver	(Inherited from System.Windows.UIElement)
	Background	(Inherited from System.Windows.Controls.Control)
	BindingGroup	(Inherited from System.Windows.FrameworkElement)

	BitmapEffect	(Inherited from System.Windows.UIElement)
	BitmapEffectInput	(Inherited from System.Windows.UIElement)
	BorderBrush	(Inherited from System.Windows.Controls.Control)
	BorderThickness	(Inherited from System.Windows.Controls.Control)
	CacheMode	(Inherited from System.Windows.UIElement)
	ClientCache	Gets or sets the C1.Data.Entities.EntityClientCache used by this C1DataSource to access the data.
	ClientScope	Gets the client scope to which this C1DataSource belongs.
	Clip	(Inherited from System.Windows.UIElement)
	ClipToBounds	(Inherited from System.Windows.UIElement)
	CommandBindings	(Inherited from System.Windows.UIElement)
	ContextMenu	(Inherited from System.Windows.FrameworkElement)
	Cursor	(Inherited from System.Windows.FrameworkElement)
	DataContext	(Inherited from System.Windows.FrameworkElement)
	DependencyObjectType	(Inherited from System.Windows.DependencyObject)
	DesiredSize	(Inherited from System.Windows.UIElement)
	Dispatcher	(Inherited from System.Windows.Threading.DispatcherObject)
	Effect	(Inherited from System.Windows.UIElement)

	FlowDirection	(Inherited from System.Windows.FrameworkElement)
	Focusable	(Inherited from System.Windows.UIElement)
	FocusVisualStyle	(Inherited from System.Windows.FrameworkElement)
	FontFamily	(Inherited from System.Windows.Controls.Control)
	FontSize	(Inherited from System.Windows.Controls.Control)
	FontStretch	(Inherited from System.Windows.Controls.Control)
	FontStyle	(Inherited from System.Windows.Controls.Control)
	FontWeight	(Inherited from System.Windows.Controls.Control)
	ForceCursor	(Inherited from System.Windows.FrameworkElement)
	Foreground	(Inherited from System.Windows.Controls.Control)
	HasAnimatedProperties	(Inherited from System.Windows.UIElement)
	Height	(Inherited from System.Windows.FrameworkElement)
	HorizontalAlignment	(Inherited from System.Windows.FrameworkElement)
	HorizontalContentAlignment	(Inherited from System.Windows.Controls.Control)
	InputBindings	(Inherited from System.Windows.UIElement)
	InputScope	(Inherited from System.Windows.FrameworkElement)
	IsArrangeValid	(Inherited from System.Windows.UIElement)
	IsEnabled	(Inherited from System.Windows.UIElement)

	IsFocused	(Inherited from System.Windows.UIElement)
	IsHitTestVisible	(Inherited from System.Windows.UIElement)
	IsInitialized	(Inherited from System.Windows.FrameworkElement)
	IsInputMethodEnabled	(Inherited from System.Windows.UIElement)
	IsKeyboardFocused	(Inherited from System.Windows.UIElement)
	IsKeyboardFocusWithin	(Inherited from System.Windows.UIElement)
	IsLoaded	(Inherited from System.Windows.FrameworkElement)
	IsManipulationEnabled	(Inherited from System.Windows.UIElement)
	IsMeasureValid	(Inherited from System.Windows.UIElement)
	IsMouseCaptured	(Inherited from System.Windows.UIElement)
	IsMouseCaptureWithin	(Inherited from System.Windows.UIElement)
	IsMouseDirectlyOver	(Inherited from System.Windows.UIElement)
	IsMouseOver	(Inherited from System.Windows.UIElement)
	IsSealed	(Inherited from System.Windows.DependencyObject)
	IsStylusCaptured	(Inherited from System.Windows.UIElement)
	IsStylusCaptureWithin	(Inherited from System.Windows.UIElement)
	IsStylusDirectlyOver	(Inherited from System.Windows.UIElement)
	IsStylusOver	(Inherited from System.Windows.UIElement)

	IsTabStop	(Inherited from System.Windows.Controls.Control)
	IsVisible	(Inherited from System.Windows.UIElement)
	Item	Overloaded. Gets the C1.Data.DataSource.ClientCollectionView of the C1.Data.Entities.EntityViewSource with the specified name in the ViewSources collection.
	Language	(Inherited from System.Windows.FrameworkElement)
	LayoutTransform	(Inherited from System.Windows.FrameworkElement)
	Margin	(Inherited from System.Windows.FrameworkElement)
	MaxHeight	(Inherited from System.Windows.FrameworkElement)
	MaxWidth	(Inherited from System.Windows.FrameworkElement)
	MinHeight	(Inherited from System.Windows.FrameworkElement)
	MinWidth	(Inherited from System.Windows.FrameworkElement)
	Name	(Inherited from System.Windows.FrameworkElement)
	ObjectContext	Gets the System.Data.Objects.ObjectContext the ClientCache is connected to.
	ObjectContextType	Gets or sets the type of an System.Data.Objects.ObjectContext used to obtain the default client cache.
	Opacity	(Inherited from System.Windows.UIElement)
	OpacityMask	(Inherited from System.Windows.UIElement)
	OverridesDefaultStyle	(Inherited from System.Windows.FrameworkElement)

	Padding	(Inherited from System.Windows.Controls.Control)
	Parent	(Inherited from System.Windows.FrameworkElement)
	PersistId	(Inherited from System.Windows.UIElement)
	RefreshInterval	Gets or sets the interval between automatic Refresh operations to refresh the data with any changes that may have occurred on the server.
	RenderSize	(Inherited from System.Windows.UIElement)
	RenderTransform	(Inherited from System.Windows.UIElement)
	RenderTransformOrigin	(Inherited from System.Windows.UIElement)
	Resources	(Inherited from System.Windows.FrameworkElement)
	SnapsToDevicePixels	(Inherited from System.Windows.UIElement)
	Style	(Inherited from System.Windows.FrameworkElement)
	TabIndex	(Inherited from System.Windows.Controls.Control)
	Tag	(Inherited from System.Windows.FrameworkElement)
	Template	(Inherited from System.Windows.Controls.Control)
	TemplatedParent	(Inherited from System.Windows.FrameworkElement)
	ToolTip	(Inherited from System.Windows.FrameworkElement)
	TouchesCaptured	(Inherited from System.Windows.UIElement)
	TouchesCapturedWithin	(Inherited from System.Windows.UIElement)

	TouchesDirectlyOver	(Inherited from System.Windows.UIElement)
	TouchesOver	(Inherited from System.Windows.UIElement)
	Triggers	(Inherited from System.Windows.FrameworkElement)
	Uid	(Inherited from System.Windows.UIElement)
	UseLayoutRounding	(Inherited from System.Windows.FrameworkElement)
	VerticalAlignment	(Inherited from System.Windows.FrameworkElement)
	VerticalContentAlignment	(Inherited from System.Windows.Controls.Control)
	ViewSources	Gets a collection of C1.Data.Entities.EntityViewSource objects that define views (based on queries) in this C1DataSource.
	Visibility	(Inherited from System.Windows.UIElement)
	Width	(Inherited from System.Windows.FrameworkElement)

Top

Public Methods

	Name	Description
	AddHandler	Overloaded. (Inherited from System.Windows.UIElement)
	AddToEventRoute	(Inherited from System.Windows.UIElement)
	ApplyAnimationClock	Overloaded. (Inherited from System.Windows.UIElement)
	ApplyTemplate	(Inherited from System.Windows.FrameworkElement)
	Arrange	(Inherited from System.Windows.UIElement)

	BeginAnimation	Overloaded. (Inherited from System.Windows.UIElement)
	BeginInit	(Inherited from System.Windows.FrameworkElement)
	BeginStoryboard	Overloaded. (Inherited from System.Windows.FrameworkElement)
	BringIntoView	Overloaded. (Inherited from System.Windows.FrameworkElement)
	CaptureMouse	(Inherited from System.Windows.UIElement)
	CaptureStylus	(Inherited from System.Windows.UIElement)
	CaptureTouch	(Inherited from System.Windows.UIElement)
	ClearValue	Overloaded. (Inherited from System.Windows.DependencyObject)
	CoerceValue	(Inherited from System.Windows.DependencyObject)
	EndInit	(Inherited from System.Windows.FrameworkElement)
	Equals	(Inherited from System.Windows.DependencyObject)
	FindCommonVisualAncestor	(Inherited from System.Windows.Media.Visual)
	FindName	(Inherited from System.Windows.FrameworkElement)
	FindResource	(Inherited from System.Windows.FrameworkElement)
	Focus	(Inherited from System.Windows.UIElement)
	GetAnimationBaseValue	(Inherited from System.Windows.UIElement)

	GetBindingExpression	(Inherited from System.Windows.FrameworkElement)
S	GetControlHandler	Gets the value of the C1.WPF.Data.Entities.C1DataSource.ControlHandler attached property from a given control.
	GetHashCode	(Inherited from System.Windows.DependencyObject)
	GetLocalValueEnumerator	(Inherited from System.Windows.DependencyObject)
	GetValue	(Inherited from System.Windows.DependencyObject)
	InputHitTest	(Inherited from System.Windows.UIElement)
	InvalidateArrange	(Inherited from System.Windows.UIElement)
	InvalidateMeasure	(Inherited from System.Windows.UIElement)
	InvalidateProperty	(Inherited from System.Windows.DependencyObject)
	InvalidateVisual	(Inherited from System.Windows.UIElement)
	IsAncestorOf	(Inherited from System.Windows.Media.Visual)
	IsDescendantOf	(Inherited from System.Windows.Media.Visual)
	Load	Loads all C1.Data.Entities.EntityViewSource objects in the ViewSources collection.
	Measure	(Inherited from System.Windows.UIElement)
	MoveFocus	(Inherited from System.Windows.FrameworkElement)
	OnApplyTemplate	(Inherited from System.Windows.FrameworkElement)

	PointFromScreen	(Inherited from System.Windows.Media.Visual)
	PointToScreen	(Inherited from System.Windows.Media.Visual)
	PredictFocus	(Inherited from System.Windows.FrameworkElement)
	RaiseEvent	(Inherited from System.Windows.UIElement)
	ReadLocalValue	(Inherited from System.Windows.DependencyObject)
	Refresh	Refreshes all C1.Data.Entities.EntityViewSource objects in the ViewSources collection.
	RegisterName	(Inherited from System.Windows.FrameworkElement)
	RejectChanges	Rejects the changes for every entity in the ObjectContext.
	ReleaseAllTouchCaptures	(Inherited from System.Windows.UIElement)
	ReleaseMouseCapture	(Inherited from System.Windows.UIElement)
	ReleaseStylusCapture	(Inherited from System.Windows.UIElement)
	ReleaseTouchCapture	(Inherited from System.Windows.UIElement)
	RemoveHandler	(Inherited from System.Windows.UIElement)
	SaveChanges	Persists all changes to the server.
	SetBinding	Overloaded. (Inherited from System.Windows.FrameworkElement)
S	SetControlHandler	Sets the value of the C1.WPF.Data.Entities.C1DataSource.ControlHandler attached property to a given control.

	SetCurrentValue	(Inherited from System.Windows.DependencyObject)
	SetResourceReference	(Inherited from System.Windows.FrameworkElement)
	SetValue	Overloaded. (Inherited from System.Windows.DependencyObject)
	ToString	(Inherited from System.Windows.Controls.Control)
	TransformToAncestor	Overloaded. (Inherited from System.Windows.Media.Visual)
	TransformToDescendant	(Inherited from System.Windows.Media.Visual)
	TransformToVisual	(Inherited from System.Windows.Media.Visual)
	TranslatePoint	(Inherited from System.Windows.UIElement)
	TryFindResource	(Inherited from System.Windows.FrameworkElement)
	UnregisterName	(Inherited from System.Windows.FrameworkElement)
	UpdateLayout	(Inherited from System.Windows.UIElement)

Top

Public Events

	Name	Description
	ContextMenuClosing	(Inherited from System.Windows.FrameworkElement)
	ContextMenuOpening	(Inherited from System.Windows.FrameworkElement)
	DataContextChanged	(Inherited from System.Windows.FrameworkElement)
	DragEnter	(Inherited from System.Windows.UIElement)

	DragLeave	(Inherited from System.Windows.UIElement)
	DragOver	(Inherited from System.Windows.UIElement)
	Drop	(Inherited from System.Windows.UIElement)
	FocusableChanged	(Inherited from System.Windows.UIElement)
	GiveFeedback	(Inherited from System.Windows.UIElement)
	GotFocus	(Inherited from System.Windows.UIElement)
	GotKeyboardFocus	(Inherited from System.Windows.UIElement)
	GotMouseCapture	(Inherited from System.Windows.UIElement)
	GotStylusCapture	(Inherited from System.Windows.UIElement)
	GotTouchCapture	(Inherited from System.Windows.UIElement)
	Initialized	(Inherited from System.Windows.FrameworkElement)
	IsEnabledChanged	(Inherited from System.Windows.UIElement)
	IsHitTestVisibleChanged	(Inherited from System.Windows.UIElement)
	IsKeyboardFocusedChanged	(Inherited from System.Windows.UIElement)
	IsKeyboardFocusWithinChanged	(Inherited from System.Windows.UIElement)
	IsMouseCapturedChanged	(Inherited from System.Windows.UIElement)
	IsMouseCaptureWithinChanged	(Inherited from System.Windows.UIElement)
	IsMouseDirectlyOverChanged	(Inherited from System.Windows.UIElement)

	IsStylusCapturedChanged	(Inherited from System.Windows.UIElement)
	IsStylusCaptureWithinChanged	(Inherited from System.Windows.UIElement)
	IsStylusDirectlyOverChanged	(Inherited from System.Windows.UIElement)
	IsVisibleChanged	(Inherited from System.Windows.UIElement)
	KeyDown	(Inherited from System.Windows.UIElement)
	KeyUp	(Inherited from System.Windows.UIElement)
	LayoutUpdated	(Inherited from System.Windows.UIElement)
	Loaded	(Inherited from System.Windows.FrameworkElement)
	LostFocus	(Inherited from System.Windows.UIElement)
	LostKeyboardFocus	(Inherited from System.Windows.UIElement)
	LostMouseCapture	(Inherited from System.Windows.UIElement)
	LostStylusCapture	(Inherited from System.Windows.UIElement)
	LostTouchCapture	(Inherited from System.Windows.UIElement)
	ManipulationBoundaryFeedback	(Inherited from System.Windows.UIElement)
	ManipulationCompleted	(Inherited from System.Windows.UIElement)
	ManipulationDelta	(Inherited from System.Windows.UIElement)
	ManipulationInertiaStarting	(Inherited from System.Windows.UIElement)
	ManipulationStarted	(Inherited from System.Windows.UIElement)

	ManipulationStarting	(Inherited from System.Windows.UIElement)
	MouseDoubleClick	(Inherited from System.Windows.Controls.Control)
	MouseDown	(Inherited from System.Windows.UIElement)
	MouseEnter	(Inherited from System.Windows.UIElement)
	MouseLeave	(Inherited from System.Windows.UIElement)
	MouseLeftButtonDown	(Inherited from System.Windows.UIElement)
	MouseLeftButtonUp	(Inherited from System.Windows.UIElement)
	MouseMove	(Inherited from System.Windows.UIElement)
	MouseRightButtonDown	(Inherited from System.Windows.UIElement)
	MouseRightButtonUp	(Inherited from System.Windows.UIElement)
	MouseUp	(Inherited from System.Windows.UIElement)
	MouseWheel	(Inherited from System.Windows.UIElement)
	PreviewDragEnter	(Inherited from System.Windows.UIElement)
	PreviewDragLeave	(Inherited from System.Windows.UIElement)
	PreviewDragOver	(Inherited from System.Windows.UIElement)
	PreviewDrop	(Inherited from System.Windows.UIElement)
	PreviewGiveFeedback	(Inherited from System.Windows.UIElement)
	PreviewGotKeyboardFocus	(Inherited from System.Windows.UIElement)

	PreviewKeyDown	(Inherited from System.Windows.UIElement)
	PreviewKeyUp	(Inherited from System.Windows.UIElement)
	PreviewLostKeyboardFocus	(Inherited from System.Windows.UIElement)
	PreviewMouseDoubleClick	(Inherited from System.Windows.Controls.Control)
	PreviewMouseDown	(Inherited from System.Windows.UIElement)
	PreviewMouseLeftButtonDown	(Inherited from System.Windows.UIElement)
	PreviewMouseLeftButtonUp	(Inherited from System.Windows.UIElement)
	PreviewMouseMove	(Inherited from System.Windows.UIElement)
	PreviewMouseRightButtonDown	(Inherited from System.Windows.UIElement)
	PreviewMouseRightButtonUp	(Inherited from System.Windows.UIElement)
	PreviewMouseUp	(Inherited from System.Windows.UIElement)
	PreviewMouseWheel	(Inherited from System.Windows.UIElement)
	PreviewQueryContinueDrag	(Inherited from System.Windows.UIElement)
	PreviewStylusButtonDown	(Inherited from System.Windows.UIElement)
	PreviewStylusButtonUp	(Inherited from System.Windows.UIElement)
	PreviewStylusDown	(Inherited from System.Windows.UIElement)
	PreviewStylusInAirMove	(Inherited from System.Windows.UIElement)
	PreviewStylusInRange	(Inherited from System.Windows.UIElement)

	PreviewStylusMove	(Inherited from System.Windows.UIElement)
	PreviewStylusOutOfRange	(Inherited from System.Windows.UIElement)
	PreviewStylusSystemGesture	(Inherited from System.Windows.UIElement)
	PreviewStylusUp	(Inherited from System.Windows.UIElement)
	PreviewTextInput	(Inherited from System.Windows.UIElement)
	PreviewTouchDown	(Inherited from System.Windows.UIElement)
	PreviewTouchMove	(Inherited from System.Windows.UIElement)
	PreviewTouchUp	(Inherited from System.Windows.UIElement)
	QueryContinueDrag	(Inherited from System.Windows.UIElement)
	QueryCursor	(Inherited from System.Windows.UIElement)
	RequestBringIntoView	(Inherited from System.Windows.FrameworkElement)
	SavedChanges	Occurs after a save operation is completed.
	SavingChanges	Occurs before changes are saved.
	SizeChanged	(Inherited from System.Windows.FrameworkElement)
	SourceUpdated	(Inherited from System.Windows.FrameworkElement)
	StylusButtonDown	(Inherited from System.Windows.UIElement)
	StylusButtonUp	(Inherited from System.Windows.UIElement)
	StylusDown	(Inherited from System.Windows.UIElement)

	StylusEnter	(Inherited from System.Windows.UIElement)
	StylusInAirMove	(Inherited from System.Windows.UIElement)
	StylusInRange	(Inherited from System.Windows.UIElement)
	StylusLeave	(Inherited from System.Windows.UIElement)
	StylusMove	(Inherited from System.Windows.UIElement)
	StylusOutOfRange	(Inherited from System.Windows.UIElement)
	StylusSystemGesture	(Inherited from System.Windows.UIElement)
	StylusUp	(Inherited from System.Windows.UIElement)
	TargetUpdated	(Inherited from System.Windows.FrameworkElement)
	TextInput	(Inherited from System.Windows.UIElement)
	ToolTipClosing	(Inherited from System.Windows.FrameworkElement)
	ToolTipOpening	(Inherited from System.Windows.FrameworkElement)
	TouchDown	(Inherited from System.Windows.UIElement)
	TouchEnter	(Inherited from System.Windows.UIElement)
	TouchLeave	(Inherited from System.Windows.UIElement)
	TouchMove	(Inherited from System.Windows.UIElement)
	TouchUp	(Inherited from System.Windows.UIElement)
	Unloaded	(Inherited from System.Windows.FrameworkElement)

Top

See Also

Reference

C1DataSource Class
C1.WPF.Data.Entities Namespace

C1DataSource Constructor
Initializes a new instance of the C1DataSource class.

Syntax

Visual Basic (Declaration)
`Public Function New()`
C#
`public C1DataSource()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

Methods
For a list of all members of this type, see C1DataSource members.

Public Methods

	Name	Description
	AddHandler	Overloaded. (Inherited from System.Windows.UIElement)

	AddToEventRoute	(Inherited from System.Windows.UIElement)
	ApplyAnimationClock	Overloaded. (Inherited from System.Windows.UIElement)
	ApplyTemplate	(Inherited from System.Windows.FrameworkElement)
	Arrange	(Inherited from System.Windows.UIElement)
	BeginAnimation	Overloaded. (Inherited from System.Windows.UIElement)
	BeginInit	(Inherited from System.Windows.FrameworkElement)
	BeginStoryboard	Overloaded. (Inherited from System.Windows.FrameworkElement)
	BringIntoView	Overloaded. (Inherited from System.Windows.FrameworkElement)
	CaptureMouse	(Inherited from System.Windows.UIElement)
	CaptureStylus	(Inherited from System.Windows.UIElement)
	CaptureTouch	(Inherited from System.Windows.UIElement)
	ClearValue	Overloaded. (Inherited from System.Windows.DependencyObject)
	CoerceValue	(Inherited from System.Windows.DependencyObject)
	EndInit	(Inherited from System.Windows.FrameworkElement)
	Equals	(Inherited from System.Windows.DependencyObject)
	FindCommonVisualAncestor	(Inherited from System.Windows.Media.Visual)

	FindName	(Inherited from System.Windows.FrameworkElement)
	FindResource	(Inherited from System.Windows.FrameworkElement)
	Focus	(Inherited from System.Windows.UIElement)
	GetAnimationBaseValue	(Inherited from System.Windows.UIElement)
	GetBindingExpression	(Inherited from System.Windows.FrameworkElement)
S	GetControlHandler	Gets the value of the C1.WPF.Data.Entities.C1DataSource.ControlHandler attached property from a given control.
	GetHashCode	(Inherited from System.Windows.DependencyObject)
	GetLocalValueEnumerator	(Inherited from System.Windows.DependencyObject)
	GetValue	(Inherited from System.Windows.DependencyObject)
	InputHitTest	(Inherited from System.Windows.UIElement)
	InvalidateArrange	(Inherited from System.Windows.UIElement)
	InvalidateMeasure	(Inherited from System.Windows.UIElement)
	InvalidateProperty	(Inherited from System.Windows.DependencyObject)
	InvalidateVisual	(Inherited from System.Windows.UIElement)
	IsAncestorOf	(Inherited from System.Windows.Media.Visual)
	IsDescendantOf	(Inherited from System.Windows.Media.Visual)
	Load	Loads all C1.Data.Entities.EntityViewSource objects in the

		ViewSources collection.
	Measure	(Inherited from System.Windows.UIElement)
	MoveFocus	(Inherited from System.Windows.FrameworkElement)
	OnApplyTemplate	(Inherited from System.Windows.FrameworkElement)
	PointFromScreen	(Inherited from System.Windows.Media.Visual)
	PointToScreen	(Inherited from System.Windows.Media.Visual)
	PredictFocus	(Inherited from System.Windows.FrameworkElement)
	RaiseEvent	(Inherited from System.Windows.UIElement)
	ReadLocalValue	(Inherited from System.Windows.DependencyObject)
	Refresh	Refreshes all C1.Data.Entities.EntityViewSource objects in the ViewSources collection.
	RegisterName	(Inherited from System.Windows.FrameworkElement)
	RejectChanges	Rejects the changes for every entity in the ObjectContext.
	ReleaseAllTouchCaptures	(Inherited from System.Windows.UIElement)
	ReleaseMouseCapture	(Inherited from System.Windows.UIElement)
	ReleaseStylusCapture	(Inherited from System.Windows.UIElement)
	ReleaseTouchCapture	(Inherited from System.Windows.UIElement)
	RemoveHandler	(Inherited from System.Windows.UIElement)

	SaveChanges	Persists all changes to the server.
	SetBinding	Overloaded. (Inherited from System.Windows.FrameworkElement)
S	SetControlHandler	Sets the value of the C1.WPF.Data.Entities.C1DataSource.ControlHandler attached property to a given control.
	SetCurrentValue	(Inherited from System.Windows.DependencyObject)
	SetResourceReference	(Inherited from System.Windows.FrameworkElement)
	SetValue	Overloaded. (Inherited from System.Windows.DependencyObject)
	ToString	(Inherited from System.Windows.Controls.Control)
	TransformToAncestor	Overloaded. (Inherited from System.Windows.Media.Visual)
	TransformToDescendant	(Inherited from System.Windows.Media.Visual)
	TransformToVisual	(Inherited from System.Windows.Media.Visual)
	TranslatePoint	(Inherited from System.Windows.UIElement)
	TryFindResource	(Inherited from System.Windows.FrameworkElement)
	UnregisterName	(Inherited from System.Windows.FrameworkElement)
	UpdateLayout	(Inherited from System.Windows.UIElement)

Top

See Also

Reference

C1DataSource Class
C1.WPF.Data.Entities Namespace

GetControlHandler Method
The control from which to read the property value.

Gets the value of the C1.WPF.Data.Entities.C1DataSource.ControlHandler attached property from a given control.

Syntax

Visual Basic (Declaration)

```
Public Shared Function GetControlHandler( _
   ByVal control As DependencyObject _
) As BaseControlHandler
```

C#

```
public static BaseControlHandler GetControlHandler(
   DependencyObject control
)
```

Parameters

control

The control from which to read the property value.

Return Value

The value of the C1.WPF.Data.Entities.C1DataSource.ControlHandler attached property.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members
ControlHandlerProperty Field

Load Method

Loads all C1.Data.Entities.EntityViewSource objects in the ViewSources collection.

Syntax

Visual Basic (Declaration)
`Public Sub Load()`
C#
`public void Load()`

Remarks

This method calls C1.Data.DataSource.ClientViewSource.Load for all elements of the ViewSources collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

Refresh Method

Refreshes all C1.Data.Entities.EntityViewSource objects in the ViewSources collection.

Syntax

Visual Basic (Declaration)
`Public Sub Refresh()`

C#

```
public void Refresh()
```

Remarks

This method calls C1.Data.DataSource.ClientViewSource.Refresh for all elements of the ViewSources collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

RejectChanges Method
Rejects the changes for every entity in the ObjectContext.

Syntax

Visual Basic (Declaration)

```
Public Sub RejectChanges()
```

C#

```
public void RejectChanges()
```

Remarks

Changes will be rejected for all entities in the ObjectContext, including those that were not loaded through this C1DataSource. This will also cancel a pending Add or Edit in the collection views.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

SaveChanges Method
Persists all changes to the server.

Syntax

Visual Basic (Declaration)	
`Public Sub SaveChanges()`	
C#	
`public void SaveChanges()`	

Remarks

Changes will be saved for all entities in the ObjectContext, including those that were not loaded through this C1DataSource. This will also commit a pending Add or Edit in the collection views.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

SetControlHandler Method
The object on which to set the C1.WPF.Data.Entities.C1DataSource.ControlHandler attached property.

The property value to set.

Sets the value of the C1.WPF.Data.Entities.C1DataSource.ControlHandler attached property to a given control.

Syntax

Visual Basic (Declaration)

```
Public Shared Sub SetControlHandler( _
   ByVal control As DependencyObject, _
   ByVal handler As BaseControlHandler _
)
```

C#

```
public static void SetControlHandler(
   DependencyObject control,
   BaseControlHandler handler
)
```

Parameters

control

The object on which to set the C1.WPF.Data.Entities.C1DataSource.ControlHandler attached property.

handler

The property value to set.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members
ControlHandlerProperty Field

Properties

For a list of all members of this type, see C1DataSource members.

Public Properties

	Name	Description
	ActualHeight	(Inherited from System.Windows.FrameworkElement)
	ActualWidth	(Inherited from System.Windows.FrameworkElement)
	AllowDrop	(Inherited from System.Windows.UIElement)
	AreAnyTouchesCaptured	(Inherited from System.Windows.UIElement)
	AreAnyTouchesCapturedWithin	(Inherited from System.Windows.UIElement)
	AreAnyTouchesDirectlyOver	(Inherited from System.Windows.UIElement)
	AreAnyTouchesOver	(Inherited from System.Windows.UIElement)
	Background	(Inherited from System.Windows.Controls.Control)
	BindingGroup	(Inherited from System.Windows.FrameworkElement)
	BitmapEffect	(Inherited from System.Windows.UIElement)
	BitmapEffectInput	(Inherited from System.Windows.UIElement)
	BorderBrush	(Inherited from System.Windows.Controls.Control)
	BorderThickness	(Inherited from System.Windows.Controls.Control)
	CacheMode	(Inherited from System.Windows.UIElement)
	ClientCache	Gets or sets the C1.Data.Entities.EntityClientCache used by this

		C1DataSource to access the data.
	ClientScope	Gets the client scope to which this C1DataSource belongs.
	Clip	(Inherited from System.Windows.UIElement)
	ClipToBounds	(Inherited from System.Windows.UIElement)
	CommandBindings	(Inherited from System.Windows.UIElement)
	ContextMenu	(Inherited from System.Windows.FrameworkElement)
	Cursor	(Inherited from System.Windows.FrameworkElement)
	DataContext	(Inherited from System.Windows.FrameworkElement)
	DependencyObjectType	(Inherited from System.Windows.DependencyObject)
	DesiredSize	(Inherited from System.Windows.UIElement)
	Dispatcher	(Inherited from System.Windows.Threading.DispatcherObject)
	Effect	(Inherited from System.Windows.UIElement)
	FlowDirection	(Inherited from System.Windows.FrameworkElement)
	Focusable	(Inherited from System.Windows.UIElement)
	FocusVisualStyle	(Inherited from System.Windows.FrameworkElement)
	FontFamily	(Inherited from System.Windows.Controls.Control)
	FontSize	(Inherited from System.Windows.Controls.Control)
	FontStretch	(Inherited from System.Windows.Controls.Control)

	FontStyle	(Inherited from System.Windows.Controls.Control)
	FontWeight	(Inherited from System.Windows.Controls.Control)
	ForceCursor	(Inherited from System.Windows.FrameworkElement)
	Foreground	(Inherited from System.Windows.Controls.Control)
	HasAnimatedProperties	(Inherited from System.Windows.UIElement)
	Height	(Inherited from System.Windows.FrameworkElement)
	HorizontalAlignment	(Inherited from System.Windows.FrameworkElement)
	HorizontalContentAlignment	(Inherited from System.Windows.Controls.Control)
	InputBindings	(Inherited from System.Windows.UIElement)
	InputScope	(Inherited from System.Windows.FrameworkElement)
	IsArrangeValid	(Inherited from System.Windows.UIElement)
	IsEnabled	(Inherited from System.Windows.UIElement)
	IsFocused	(Inherited from System.Windows.UIElement)
	IsHitTestVisible	(Inherited from System.Windows.UIElement)
	IsInitialized	(Inherited from System.Windows.FrameworkElement)
	IsInputMethodEnabled	(Inherited from System.Windows.UIElement)
	IsKeyboardFocused	(Inherited from System.Windows.UIElement)
	IsKeyboardFocusWithin	(Inherited from System.Windows.UIElement)

	IsLoaded	(Inherited from System.Windows.FrameworkElement)
	IsManipulationEnabled	(Inherited from System.Windows.UIElement)
	IsMeasureValid	(Inherited from System.Windows.UIElement)
	IsMouseCaptured	(Inherited from System.Windows.UIElement)
	IsMouseCaptureWithin	(Inherited from System.Windows.UIElement)
	IsMouseDirectlyOver	(Inherited from System.Windows.UIElement)
	IsMouseOver	(Inherited from System.Windows.UIElement)
	IsSealed	(Inherited from System.Windows.DependencyObject)
	IsStylusCaptured	(Inherited from System.Windows.UIElement)
	IsStylusCaptureWithin	(Inherited from System.Windows.UIElement)
	IsStylusDirectlyOver	(Inherited from System.Windows.UIElement)
	IsStylusOver	(Inherited from System.Windows.UIElement)
	IsTabStop	(Inherited from System.Windows.Controls.Control)
	IsVisible	(Inherited from System.Windows.UIElement)
	Item	Overloaded. Gets the C1.Data.DataSource.ClientCollectionView of the C1.Data.Entities.EntityViewSource with the specified name in the ViewSources collection.
	Language	(Inherited from System.Windows.FrameworkElement)
	LayoutTransform	(Inherited from System.Windows.FrameworkElement)

	Margin	(Inherited from System.Windows.FrameworkElement)
	MaxHeight	(Inherited from System.Windows.FrameworkElement)
	MaxWidth	(Inherited from System.Windows.FrameworkElement)
	MinHeight	(Inherited from System.Windows.FrameworkElement)
	MinWidth	(Inherited from System.Windows.FrameworkElement)
	Name	(Inherited from System.Windows.FrameworkElement)
	ObjectContext	Gets the System.Data.Objects.ObjectContext the ClientCache is connected to.
	ObjectContextType	Gets or sets the type of an System.Data.Objects.ObjectContext used to obtain the default client cache.
	Opacity	(Inherited from System.Windows.UIElement)
	OpacityMask	(Inherited from System.Windows.UIElement)
	OverridesDefaultStyle	(Inherited from System.Windows.FrameworkElement)
	Padding	(Inherited from System.Windows.Controls.Control)
	Parent	(Inherited from System.Windows.FrameworkElement)
	PersistId	(Inherited from System.Windows.UIElement)
	RefreshInterval	Gets or sets the interval between automatic Refresh operations to refresh the data with any changes that may have occurred on the server.
	RenderSize	(Inherited from System.Windows.UIElement)

	RenderTransform	(Inherited from System.Windows.UIElement)
	RenderTransformOrigin	(Inherited from System.Windows.UIElement)
	Resources	(Inherited from System.Windows.FrameworkElement)
	SnapsToDevicePixels	(Inherited from System.Windows.UIElement)
	Style	(Inherited from System.Windows.FrameworkElement)
	TabIndex	(Inherited from System.Windows.Controls.Control)
	Tag	(Inherited from System.Windows.FrameworkElement)
	Template	(Inherited from System.Windows.Controls.Control)
	TemplatedParent	(Inherited from System.Windows.FrameworkElement)
	ToolTip	(Inherited from System.Windows.FrameworkElement)
	TouchesCaptured	(Inherited from System.Windows.UIElement)
	TouchesCapturedWithin	(Inherited from System.Windows.UIElement)
	TouchesDirectlyOver	(Inherited from System.Windows.UIElement)
	TouchesOver	(Inherited from System.Windows.UIElement)
	Triggers	(Inherited from System.Windows.FrameworkElement)
	Uid	(Inherited from System.Windows.UIElement)
	UseLayoutRounding	(Inherited from System.Windows.FrameworkElement)
	VerticalAlignment	(Inherited from System.Windows.FrameworkElement)

	VerticalContentAlignment	(Inherited from System.Windows.Controls.Control)
	ViewSources	Gets a collection of C1.Data.Entities.EntityViewSource objects that define views (based on queries) in this C1DataSource.
	Visibility	(Inherited from System.Windows.UIElement)
	Width	(Inherited from System.Windows.FrameworkElement)

Top

See Also

Reference

C1DataSource Class
C1.WPF.Data.Entities Namespace

ClientCache Property
Gets or sets the C1.Data.Entities.EntityClientCache used by this C1DataSource to access the data.

Syntax

Visual Basic (Declaration)

```
Public Property ClientCache As EntityClientCache
```

C#

```
public EntityClientCache ClientCache {get; set;}
```

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

ClientScope Property

Gets the client scope to which this C1DataSource belongs.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property ClientScope As EntityClientScope`
C#
`public EntityClientScope ClientScope {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

Item Property

Gets the C1.Data.DataSource.ClientCollectionView of the C1.Data.Entities.EntityViewSource with the specified name in the ViewSources collection.

Overload List

Overload	Description
Item(String)	Gets the C1.Data.DataSource.ClientCollectionView of the C1.Data.Entities.EntityViewSource with the specified name in the ViewSources collection.
Item(Int32)	Gets the C1.Data.DataSource.ClientCollectionView of the C1.Data.Entities.EntityViewSource at the specified index in the ViewSources collection.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

Item(String) Property
The name of the C1.Data.Entities.EntityViewSource to take the C1.Data.DataSource.ClientCollectionView from.

Gets the C1.Data.DataSource.ClientCollectionView of the C1.Data.Entities.EntityViewSource with the specified name in the ViewSources collection.

Syntax

Visual Basic (Declaration)

```
Public Overloads ReadOnly Property Item( _
   ByVal name As String _
) As ClientCollectionView
```

C#

```
public ClientCollectionView Item(
   string name
) {get;}
```

Parameters

name

The name of the C1.Data.Entities.EntityViewSource to take the C1.Data.DataSource.ClientCollectionView from.

Property Value

The C1.Data.DataSource.ClientCollectionView of the C1.Data.Entities.EntityViewSource.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members
Overload List

Item(Int32) Property
The index of the C1.Data.Entities.EntityViewSource to take the C1.Data.DataSource.ClientCollectionView from.

Gets the C1.Data.DataSource.ClientCollectionView of the C1.Data.Entities.EntityViewSource at the specified index in the ViewSources collection.

Syntax

Visual Basic (Declaration)

```
Public Overloads ReadOnly Property Item( _
   ByVal index As Integer _
) As ClientCollectionView
```

C#

```
public ClientCollectionView Item(
   int index
) {get;}
```

Parameters

index

The index of the C1.Data.Entities.EntityViewSource to take the C1.Data.DataSource.ClientCollectionView from.

Property Value

The C1.Data.DataSource.ClientCollectionView of the C1.Data.Entities.EntityViewSource.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members
Overload List

ObjectContext Property
Gets the System.Data.Objects.ObjectContext the ClientCache is connected to.

Syntax

Visual Basic (Declaration)
`Public ReadOnly Property ObjectContext As ObjectContext`
C#
`public ObjectContext ObjectContext {get;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

ObjectContextType Property
Gets or sets the type of an System.Data.Objects.ObjectContext used to obtain the default client cache.

Syntax

Visual Basic (Declaration)
`Public Property ObjectContextType As Type`
C#
`public Type ObjectContextType {get; set;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

RefreshInterval Property
Gets or sets the interval between automatic Refresh operations to refresh the data with any changes that may have occurred on the server.

Syntax

Visual Basic (Declaration)
`Public Property RefreshInterval As TimeSpan`
C#
`public TimeSpan RefreshInterval {get; set;}`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

ViewSources Property
Gets a collection of C1.Data.Entities.EntityViewSource objects that define views (based on queries) in this C1DataSource.

Syntax

Visual Basic (Declaration)	
`Public ReadOnly Property ViewSources As EntityViewSourceCollection`	
C#	
`public EntityViewSourceCollection ViewSources {get;}`	

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

Fields
For a list of all members of this type, see C1DataSource members.

Public Fields

	Name	Description
S	ControlHandlerProperty	Identifies the C1.WPF.Data.Entities.C1DataSource.ControlHandler attached property.

Top

See Also

Reference

C1DataSource Class
C1.WPF.Data.Entities Namespace

ControlHandlerProperty Field
Identifies the C1.WPF.Data.Entities.C1DataSource.ControlHandler attached property.

Syntax

Visual Basic (Declaration)	
`Public Shared ReadOnly ControlHandlerProperty As DependencyProperty`	
C#	
`public static readonly DependencyProperty ControlHandlerProperty`	

Remarks

Use this attached property to connect a control handler to a control.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members

Events
For a list of all members of this type, see C1DataSource members.

Public Events

	Name	Description

	ContextMenuClosing	(Inherited from System.Windows.FrameworkElement)
	ContextMenuOpening	(Inherited from System.Windows.FrameworkElement)
	DataContextChanged	(Inherited from System.Windows.FrameworkElement)
	DragEnter	(Inherited from System.Windows.UIElement)
	DragLeave	(Inherited from System.Windows.UIElement)
	DragOver	(Inherited from System.Windows.UIElement)
	Drop	(Inherited from System.Windows.UIElement)
	FocusableChanged	(Inherited from System.Windows.UIElement)
	GiveFeedback	(Inherited from System.Windows.UIElement)
	GotFocus	(Inherited from System.Windows.UIElement)
	GotKeyboardFocus	(Inherited from System.Windows.UIElement)
	GotMouseCapture	(Inherited from System.Windows.UIElement)
	GotStylusCapture	(Inherited from System.Windows.UIElement)
	GotTouchCapture	(Inherited from System.Windows.UIElement)
	Initialized	(Inherited from System.Windows.FrameworkElement)
	IsEnabledChanged	(Inherited from System.Windows.UIElement)
	IsHitTestVisibleChanged	(Inherited from System.Windows.UIElement)
	IsKeyboardFocusedChanged	(Inherited from System.Windows.UIElement)

	IsKeyboardFocusWithinChanged	(Inherited from System.Windows.UIElement)
	IsMouseCapturedChanged	(Inherited from System.Windows.UIElement)
	IsMouseCaptureWithinChanged	(Inherited from System.Windows.UIElement)
	IsMouseDirectlyOverChanged	(Inherited from System.Windows.UIElement)
	IsStylusCapturedChanged	(Inherited from System.Windows.UIElement)
	IsStylusCaptureWithinChanged	(Inherited from System.Windows.UIElement)
	IsStylusDirectlyOverChanged	(Inherited from System.Windows.UIElement)
	IsVisibleChanged	(Inherited from System.Windows.UIElement)
	KeyDown	(Inherited from System.Windows.UIElement)
	KeyUp	(Inherited from System.Windows.UIElement)
	LayoutUpdated	(Inherited from System.Windows.UIElement)
	Loaded	(Inherited from System.Windows.FrameworkElement)
	LostFocus	(Inherited from System.Windows.UIElement)
	LostKeyboardFocus	(Inherited from System.Windows.UIElement)
	LostMouseCapture	(Inherited from System.Windows.UIElement)
	LostStylusCapture	(Inherited from System.Windows.UIElement)
	LostTouchCapture	(Inherited from System.Windows.UIElement)
	ManipulationBoundaryFeedback	(Inherited from System.Windows.UIElement)

	ManipulationCompleted	(Inherited from System.Windows.UIElement)
	ManipulationDelta	(Inherited from System.Windows.UIElement)
	ManipulationInertiaStarting	(Inherited from System.Windows.UIElement)
	ManipulationStarted	(Inherited from System.Windows.UIElement)
	ManipulationStarting	(Inherited from System.Windows.UIElement)
	MouseDoubleClick	(Inherited from System.Windows.Controls.Control)
	MouseDown	(Inherited from System.Windows.UIElement)
	MouseEnter	(Inherited from System.Windows.UIElement)
	MouseLeave	(Inherited from System.Windows.UIElement)
	MouseLeftButtonDown	(Inherited from System.Windows.UIElement)
	MouseLeftButtonUp	(Inherited from System.Windows.UIElement)
	MouseMove	(Inherited from System.Windows.UIElement)
	MouseRightButtonDown	(Inherited from System.Windows.UIElement)
	MouseRightButtonUp	(Inherited from System.Windows.UIElement)
	MouseUp	(Inherited from System.Windows.UIElement)
	MouseWheel	(Inherited from System.Windows.UIElement)
	PreviewDragEnter	(Inherited from System.Windows.UIElement)
	PreviewDragLeave	(Inherited from System.Windows.UIElement)

	PreviewDragOver	(Inherited from System.Windows.UIElement)
	PreviewDrop	(Inherited from System.Windows.UIElement)
	PreviewGiveFeedback	(Inherited from System.Windows.UIElement)
	PreviewGotKeyboardFocus	(Inherited from System.Windows.UIElement)
	PreviewKeyDown	(Inherited from System.Windows.UIElement)
	PreviewKeyUp	(Inherited from System.Windows.UIElement)
	PreviewLostKeyboardFocus	(Inherited from System.Windows.UIElement)
	PreviewMouseDoubleClick	(Inherited from System.Windows.Controls.Control)
	PreviewMouseDown	(Inherited from System.Windows.UIElement)
	PreviewMouseLeftButtonDown	(Inherited from System.Windows.UIElement)
	PreviewMouseLeftButtonUp	(Inherited from System.Windows.UIElement)
	PreviewMouseMove	(Inherited from System.Windows.UIElement)
	PreviewMouseRightButtonDown	(Inherited from System.Windows.UIElement)
	PreviewMouseRightButtonUp	(Inherited from System.Windows.UIElement)
	PreviewMouseUp	(Inherited from System.Windows.UIElement)
	PreviewMouseWheel	(Inherited from System.Windows.UIElement)
	PreviewQueryContinueDrag	(Inherited from System.Windows.UIElement)
	PreviewStylusButtonDown	(Inherited from System.Windows.UIElement)

	PreviewStylusButtonUp	(Inherited from System.Windows.UIElement)
	PreviewStylusDown	(Inherited from System.Windows.UIElement)
	PreviewStylusInAirMove	(Inherited from System.Windows.UIElement)
	PreviewStylusInRange	(Inherited from System.Windows.UIElement)
	PreviewStylusMove	(Inherited from System.Windows.UIElement)
	PreviewStylusOutOfRange	(Inherited from System.Windows.UIElement)
	PreviewStylusSystemGesture	(Inherited from System.Windows.UIElement)
	PreviewStylusUp	(Inherited from System.Windows.UIElement)
	PreviewTextInput	(Inherited from System.Windows.UIElement)
	PreviewTouchDown	(Inherited from System.Windows.UIElement)
	PreviewTouchMove	(Inherited from System.Windows.UIElement)
	PreviewTouchUp	(Inherited from System.Windows.UIElement)
	QueryContinueDrag	(Inherited from System.Windows.UIElement)
	QueryCursor	(Inherited from System.Windows.UIElement)
	RequestBringIntoView	(Inherited from System.Windows.FrameworkElement)
	SavedChanges	Occurs after a save operation is completed.
	SavingChanges	Occurs before changes are saved.
	SizeChanged	(Inherited from System.Windows.FrameworkElement)

	SourceUpdated	(Inherited from System.Windows.FrameworkElement)
	StylusButtonDown	(Inherited from System.Windows.UIElement)
	StylusButtonUp	(Inherited from System.Windows.UIElement)
	StylusDown	(Inherited from System.Windows.UIElement)
	StylusEnter	(Inherited from System.Windows.UIElement)
	StylusInAirMove	(Inherited from System.Windows.UIElement)
	StylusInRange	(Inherited from System.Windows.UIElement)
	StylusLeave	(Inherited from System.Windows.UIElement)
	StylusMove	(Inherited from System.Windows.UIElement)
	StylusOutOfRange	(Inherited from System.Windows.UIElement)
	StylusSystemGesture	(Inherited from System.Windows.UIElement)
	StylusUp	(Inherited from System.Windows.UIElement)
	TargetUpdated	(Inherited from System.Windows.FrameworkElement)
	TextInput	(Inherited from System.Windows.UIElement)
	ToolTipClosing	(Inherited from System.Windows.FrameworkElement)
	ToolTipOpening	(Inherited from System.Windows.FrameworkElement)
	TouchDown	(Inherited from System.Windows.UIElement)
	TouchEnter	(Inherited from System.Windows.UIElement)

	TouchLeave	(Inherited from System.Windows.UIElement)
	TouchMove	(Inherited from System.Windows.UIElement)
	TouchUp	(Inherited from System.Windows.UIElement)
	Unloaded	(Inherited from System.Windows.FrameworkElement)

Top

See Also

Reference

C1DataSource Class
C1.WPF.Data.Entities Namespace

SavedChanges Event
Occurs after a save operation is completed.

Syntax

Visual Basic (Declaration)
`Public Event SavedChanges As EventHandler(Of SavedChangesEventArgs)`
C#
`public event EventHandler<SavedChangesEventArgs> SavedChanges`

Event Data

The event handler receives an argument of type SavedChangesEventArgs containing data related to this event. The following **SavedChangesEventArgs** properties provide information specific to this event.

Property	Description
Error	Gets a value showing the error that occurred during a save operation.
HasError	Gets a value indicating whether the save operation has failed. If true, inspect

	the Error property for details.
IsErrorHandled	Gets a value indicating whether the error has been marked as handled by calling MarkErrorAsHandled.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

C1DataSource Class
C1DataSource Members
SaveChanges Method

SavingChanges Event
Occurs before changes are saved.

Syntax

Visual Basic (Declaration)
`Public Event SavingChanges As EventHandler(Of CancelEventArgs)`
C#
`public event EventHandler<CancelEventArgs> SavingChanges`

Event Data

The event handler receives an argument of type CancelEventArgs containing data related to this event. The following **CancelEventArgs** properties provide information specific to this event.

Property	Description
Cancel	

Remarks

This event is raised from the SaveChanges method and allows a handler to cancel the operation before it begins. When a handler sets System.ComponentModel.CancelEventArgs.Cancel to True, the operation will be aborted and a subsequent SavedChanges event will not be raised.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

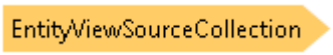
See Also

Reference

C1DataSource Class
C1DataSource Members
SaveChanges Method

EntityViewSourceCollection
An observable collection of C1.Data.Entities.EntityViewSource objects.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class EntityViewSourceCollection` `  Inherits System.Collections.ObjectModel.ObservableCollection(Of EntityViewSource)`
C#
`public class EntityViewSourceCollection : System.Collections.ObjectModel.ObservableCollection<EntityViewSource>`

Inheritance Hierarchy

System.Object
  System.Collections.ObjectModel.Collection<T>

System.Collections.ObjectModel.ObservableCollection<T>
C1.WPF.Data.Entities.EntityViewSourceCollection

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

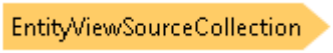
See Also

Reference

EntityViewSourceCollection Members
C1.WPF.Data.Entities Namespace

Overview
An observable collection of C1.Data.Entities.EntityViewSource objects.

Object Model



Syntax

Visual Basic (Declaration)
`Public Class EntityViewSourceCollection` `  Inherits System.Collections.ObjectModel.ObservableCollection(Of EntityViewSource)`
C#
`public class EntityViewSourceCollection : System.Collections.ObjectModel.ObservableCollection<EntityViewSource>`

Inheritance Hierarchy

System.Object
  System.Collections.ObjectModel.Collection<T>
    System.Collections.ObjectModel.ObservableCollection<T>
      C1.WPF.Data.Entities.EntityViewSourceCollection

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

EntityViewSourceCollection Members
C1.WPF.Data.Entities Namespace

Members
Properties Methods Events

The following tables list the members exposed by EntityViewSourceCollection.

Public Constructors

	Name	Description
	EntityViewSourceCollection Constructor	

Top

Public Properties

	Name	Description
	Count	(Inherited from System.Collections.ObjectModel.Collection<EntityViewSource>)
	Item	Gets the C1.Data.Entities.EntityViewSource with the specified name.

Top

Public Methods

	Name	Description

	Add	(Inherited from System.Collections.ObjectModel.Collection<EntityViewSource>)
	Clear	(Inherited from System.Collections.ObjectModel.Collection<EntityViewSource>)
	Contains	(Inherited from System.Collections.ObjectModel.Collection<EntityViewSource>)
	CopyTo	(Inherited from System.Collections.ObjectModel.Collection<EntityViewSource>)
	GetEnumerator	(Inherited from System.Collections.ObjectModel.Collection<EntityViewSource>)
	IndexOf	(Inherited from System.Collections.ObjectModel.Collection<EntityViewSource>)
	Insert	(Inherited from System.Collections.ObjectModel.Collection<EntityViewSource>)
	Move	(Inherited from System.Collections.ObjectModel.ObservableCollection<EntityViewSource>)
	Remove	(Inherited from System.Collections.ObjectModel.Collection<EntityViewSource>)
	RemoveAt	(Inherited from System.Collections.ObjectModel.Collection<EntityViewSource>)

Top

Public Events

	Name	Description

	CollectionChanged	(Inherited from System.Collections.ObjectModel.ObservableCollection<EntityViewSource>)

Top

See Also

Reference

EntityViewSourceCollection Class
C1.WPF.Data.Entities Namespace

EntityViewSourceCollection Constructor

Syntax

Visual Basic (Declaration)
`Public Function New()`
C#
`public EntityViewSourceCollection()`

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

EntityViewSourceCollection Class
EntityViewSourceCollection Members

Properties
For a list of all members of this type, see EntityViewSourceCollection members.

Public Properties

	Name	Description

	Count	(Inherited from System.Collections.ObjectModel.Collection<EntityViewSource>)
	Item	Gets the C1.Data.Entities.EntityViewSource with the specified name.

Top

See Also

Reference

EntityViewSourceCollection Class
C1.WPF.Data.Entities Namespace

Item Property
!The name of the C1.Data.Entities.EntityViewSource to get from the collection.

Gets the C1.Data.Entities.EntityViewSource with the specified name.

Syntax

Visual Basic (Declaration)

```
Public Shadows ReadOnly Default Property Item( _
   ByVal name As String _
) As EntityViewSource
```

C#

```
public new EntityViewSource this[
   string name
]; {get;}
```

Parameters

name

!The name of the C1.Data.Entities.EntityViewSource to get from the collection.

Property Value

The C1.Data.Entities.EntityViewSource with the specified name, or null if it does not exist.

Requirements

Target Platforms: Windows 7, Windows Vista SP1 or later, Windows XP SP3, Windows Server 2008 (Server Core not supported), Windows Server 2008 R2 (Server Core supported with SP1 or later), Windows Server 2003 SP2

See Also

Reference

EntityViewSourceCollection Class
EntityViewSourceCollection Members