
ComponentOne

TabControl for WPF

By GrapeCity, Inc.

Copyright © 1987-2012 GrapeCity, Inc. All rights reserved.

Corporate Headquarters

ComponentOne, a division of GrapeCity

201 South Highland Avenue
3rd 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)

Trademarks

The ComponentOne product name is a trademark and ComponentOne is a registered trademark of GrapeCity, Inc. All other trademarks used herein are the properties of their respective owners.

Warranty

ComponentOne warrants that the original CD (or diskettes) are free from defects in material and workmanship, assuming normal use, for a period of 90 days from the date of purchase. If a defect occurs during this time, you may return the defective CD (or disk) to ComponentOne, along with a dated proof of purchase, and ComponentOne will replace it at no charge. After 90 days, you can obtain a replacement for a defective CD (or disk) by sending it and a check for \$25 (to cover postage and handling) to ComponentOne.

Except for the express warranty of the original CD (or disks) set forth here, ComponentOne makes no other warranties, express or implied. Every attempt has been made to ensure that the information contained in this manual is correct as of the time it was written. We are not responsible for any errors or omissions. ComponentOne's liability is limited to the amount you paid for the product. ComponentOne is not liable for any special, consequential, or other damages for any reason.

Copying and Distribution

While you are welcome to make backup copies of the software for your own use and protection, you are not permitted to make copies for the use of anyone else. We put a lot of time and effort into creating this product, and we appreciate your support in seeing that it is used by licensed users only.

This manual was produced using [ComponentOne Doc-To-Help™](#).

Table of Contents

ComponentOne TabControl for WPF Overview	5
Help with ComponentOne Studio for WPF	5
Key Features	7
TabControl for WPF Quick Start	8
Step 1 of 4: Creating an Application with a C1TabControl Control	8
Step 2 of 4: Adding Tab Pages to the C1TabControl Control	8
Step 3 of 4: Customizing the C1TabControl Control	9
Step 4 of 4: Running the Project	10
Working with C1TabControl	11
C1TabControl Elements	12
Tabs	12
Tabstrip	13
Tab Page	13
C1TabControl Features	14
Tab Shaping	14
Tabstrip Placement	15
Tab Closing	16
Optional Tab Menu	16
Tab Overlapping	17
TabControl for WPF Layout and Appearance	17
ComponentOne ClearStyle Technology	18
How ClearStyle Works	18
C1TabControl ClearStyle Properties	18
C1TabControl Themes	19
TabControl for WPF Appearance Properties	22
TabControl Templates	24
Item Templates	25
TabControl for WPF Samples	25
TabControl for WPF Task-Based Help	27

Adding a Tab to the C1TabControl Control	27
Adding Content to a Tab Page	28
Specifying a Tab Header	29
Changing the Tabstrip Placement.....	30
Changing the Shape of Tabs	31
Allowing Users to Close Tabs.....	32
Preventing a User from Closing a Specific Tab	33
Adding a Menu to the Tabstrip.....	33
Overlapping Tabs on a Tabstrip	34

ComponentOne TabControl for WPF Overview

Arrange content in an efficient, organized manner using **ComponentOne TabControl for WPF**. The **C1TabControl** control allows you to add tabs and corresponding content pages, thus enabling you to dispense substantial amounts of information while reducing screen space usage.

For a list of the latest features added to **ComponentOne Studio for WPF**, visit [What's New in Studio for WPF](#).



Getting Started

- [Working with C1TabControl](#) (page 11)
- [Quick Start](#) (page 8)
- [Task-Based Help](#) (page 27)

Help with ComponentOne Studio for WPF

Getting Started

For information on installing **ComponentOne Studio for WPF**, licensing, technical support, namespaces and creating a project with the control, please visit [Getting Started with Studio for WPF](#).

What's New

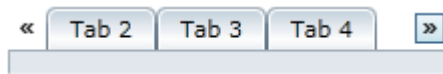
For a list of the latest features added to **ComponentOne Studio for WPF**, visit [What's New in Studio for WPF](#).

Key Features

ComponentOne TabControl for WPF allows you to create customized, rich applications. Make the most of **TabControl for WPF** by taking advantage of the following key features:

- **Scroll Elements**

The C1TabControl control will automatically insert scroll buttons when the amount of tabs exceeds the specified width or height of the control.



- **Tabstrip Placement**

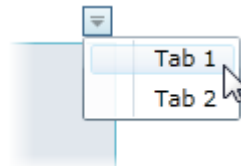
The C1TabControl control's can be placed at the top, bottom, left, or right of the control.

- **Tab Closing Options**

Control whether the user can close tabs and where to show the close button. Display the close button inside each tab item or in a global location outside the tabstrip, just like Visual Studio does in its Documents tab.

- **Show Tabs in a Menu**

The C1TabControl control has an optional tab menu that can be activated by setting the **TabStripMenuVisibility** property to **Visible**. The tab menu enables users to open tab pages using a drop-down menu.



- **Customize the Header's Shape**

You can modify the shape of the tab headers using the **TabItemShape** property in the **TabControl**; select from **Rounded**, **Rectangle**, and **Sloped**. This is ideal for non-designer users; you don't need to change the template of the control to change the shape of the tab headers.

- **New Tab Item**

C1TabControl includes built-in support for the new tab item with a uniform look and feel with the rest of the tabs, just like Microsoft Internet Explorer 8.

- **Align Items in the Header**

Define if the tab items are overlapped with the right-most in the back or the left-most in the back. The selected item is always on top.

- **Change the Background**

Keeping the appearance of the tab, you can change its background. While this seems like a very simple feature, C1TabControl is the only tab control in the market that allows you to change the background without having to customize the full template.

- **Overlap Headers**

Overlap between tab items headers can be customized to show jagged tabs, like the Documents tab in Microsoft Visual Studio.


TabControl for WPF Quick Start

The following quick start guide is intended to get you up and running with **TabControl for WPF**. In this quick start, you'll start in Blend to create a new project with the C1TabControl control. You will also customize the C1TabControl control, add tabs pages filled with content, and then observe some of the run-time features of the control.

Step 1 of 4: Creating an Application with a C1TabControl Control

In this step, you'll begin in Visual Studio to create a WPF application using **TabControl for WPF**.

Complete the following steps:

1. In Visual Studio, select **File | New | Project**.
2. In the **New Project** dialog box, select **WPF Application**.
3. Enter a **Name** and **Location** for your project and click **OK** to create the new application.
4. In the Toolbox, double-click the C1TabControl icon to add the C1TabControl control to the WPF application.
5. Add three tabs to the control by completing the following steps:
 - a. Click the C1TabControl control once to select it.
 - b. In the **Properties** window, click the **Items** ellipsis button .
The **Collection Editor: Items** dialog box opens.
 - c. Click the **Add** button three times to add three C1TabItem items to the C1TabControl control.
6. Click **OK** to close the **Collection Editor: Items** dialog box.

You have completed the first step of the **TabControl for WPF** quick start. In this step, you created a project and added a C1TabControl control with three tabs to the project. In the next step, you will customize the control's tab pages.

Step 2 of 4: Adding Tab Pages to the C1TabControl Control

In the last step, you created a WPF project in Visual Studio and then added a C1TabControl control with three tabs to it. In this step, you will customize each tab page.

Complete the following steps:

1. Switch to XAML view.
2. Add `Header="Tab 1"` to the first `<c1:C1TabItem>` tag so that the markup resembles the following:

```
<c1:C1TabItem Header="Tab 1"/>
```


3. Add `Header="Tab 2"` and `Content="I am the Content property set to a string"` to the second `<c1:C1TabItem>` tag so that the markup resembles the following:

```
<c1:C1TabItem Header="Tab 2" Content="I am the Content property set to a string"/>
```

4. Add `Header="Tab 3"`, `Content="You can't close this tab. Try it."`, and `CanUserClose="False"` to the third `<c1:C1TabItem>` tag so that the markup resembles the following:

```
<c1:C1TabItem Header="Tab 3" Content="You can't close this tab. Try it." CanUserClose="False"/>
```

Setting the `CanUserClose` property to **False** prevents users from closing the tab at run time.

5. Add a **Calendar** control as the first tab's by completing the following steps:
 - a. Switch to Design view and select the first tab.
 - b. In the Toolbox, double-click the **Calendar** icon to add the **Calendar** control to the tab.
 - c. Select the **Calendar** control and then set the following properties:
 - Set the **Height** property to "Auto".
 - Set the **Width** property to "Auto".

You have completed step 2 of 4. In this step, you customized the three pages of the `C1TabControl` control. In the next step, you'll customize the appearance and behavior of the `C1TabControl` control.

Step 3 of 4: Customizing the C1TabControl Control

In the last step, you added three customized tab pages to the `C1TabControl` control. In this step, you'll customize the `C1TabControl` control by setting several of its properties.

Complete the following steps:

1. Select the `C1TabControl` control.
2. In the **Properties** window, set the following properties:
 - Set the **Height** property to "200".
 - Set the **Width** property to "300".
 - Set the `TabItemClose` property to **InEachTab**. This will add close buttons to each tab except to the third tab, which you specified shouldn't be allowed to close in the last step of the quick start.
 - Set the `TabItemShape` property to **Sloped**. This will change the shape of the tab items so that they resemble tabs on an office folder.
 - Set the `TabStripMenuVisibility` property to **Visible**.
 - Set the `TabStripPlacement` property to **Bottom**. This will place the tabstrip at the bottom of the control.

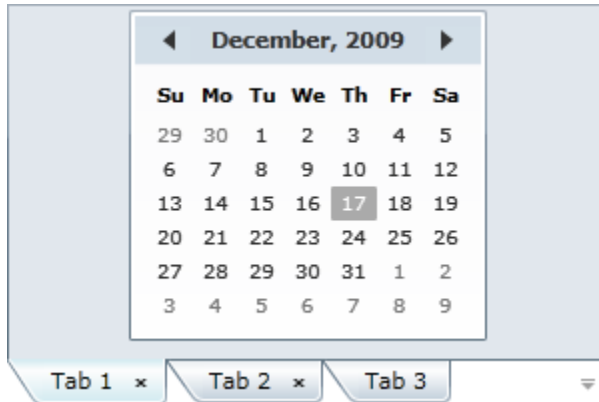
You have completed step 2 of 4 by customizing the features of the `C1TabControl` control. In the next step, you will run the program and observe what you've accomplished during this quick start.

Step 4 of 4: Running the Project

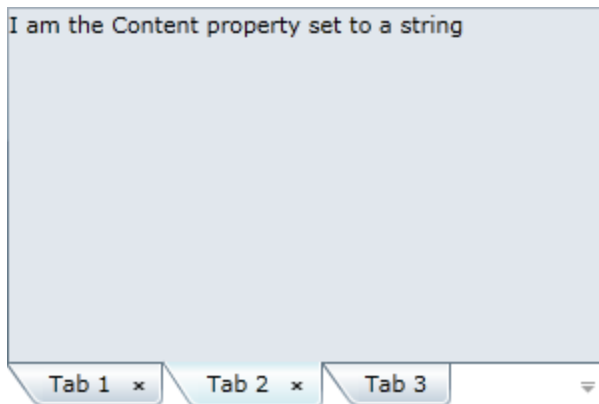
In the previous steps, you created a project with a C1TabControl control, added tab pages to the control, and modified the control's appearance and behaviors. In this step, you will run the program and observe all of the changes you made to the C1TabControl control.

Complete the following steps:

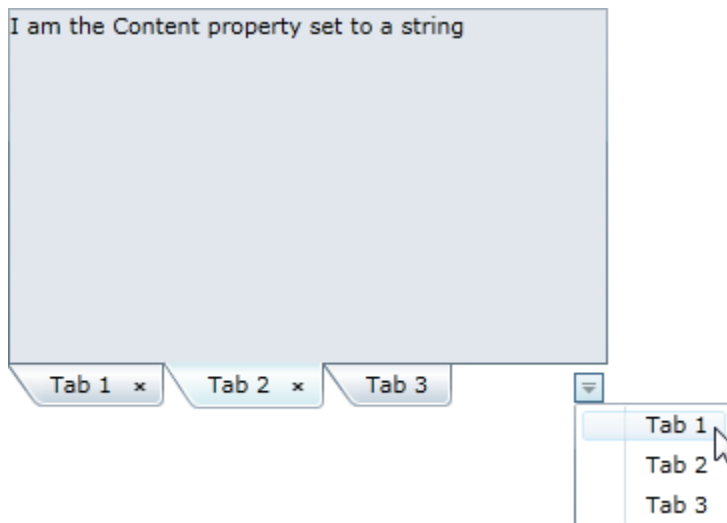
1. Press F5 to run the project. Observe that the C1TabControl control's tabstrip runs along the bottom of the control and features sloped tabs. Also note that it loads with the first tab page, which features a **Calendar** control as content, in view.



2. Click **Tab 2** and observe that the content is just the **Content** property set to a string.

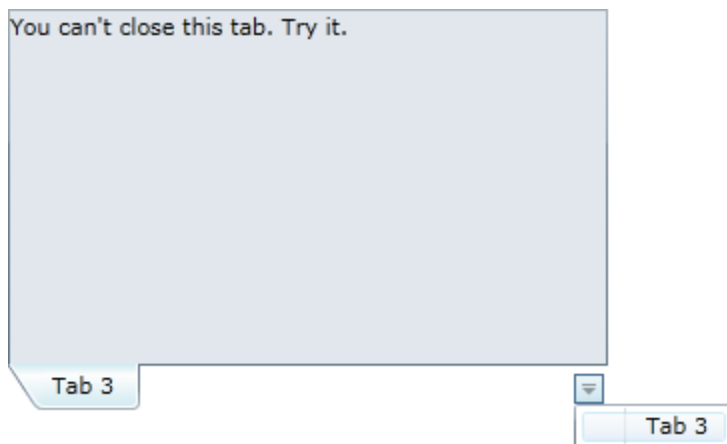


3. Click the drop-down button to open the tab menu and select **Tab 1**.



Tab 1 takes focus again.

4. Click **Tab 1**'s close button to close the tab.
5. Click **Tab 2**'s close button to close the tab. **Tab 3**'s content comes into focus; you can't close this tab because you set its `CanUserClose` property to **False** in a previous step.
6. Click the menu button and note that only the current tab page, **Tab 3**, is listed because the other two are closed.



Congratulations! You have completed all four steps of the **TabControl for WPF** quick start. In this quick start, you created a project with a fully customized `C1TabControl`. From here, you can visit [Working with C1TabControl](#) (page 11) to learn more about the control's essentials or visit [TabControl for WPF Task-Based Help](#) (page 27) to jump right into using the control.

Working with C1TabControl

The `C1TabControl` control is a container that can hold a series of tab pages. Each tab page can store text, images, and controls. Each tab and tab page is represented by the `C1TabItem` class.

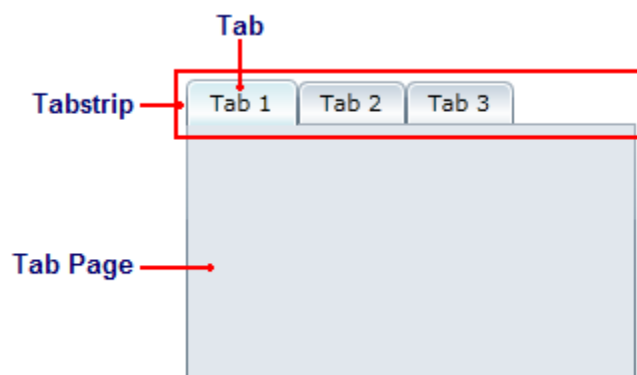
When you add the C1TabControl control to a project, it exists as nothing more than a container. But once the control is added to your project, you can easily add tabs to it in Design view, in XAML, or in code. The following image depicts a C1TabControl control with three tabs.



The following topics provide an overview of the C1TabControl control's elements and features.

C1TabControl Elements

This section provides a visual and descriptive overview of the elements that comprise the C1TabControl control. The control is comprised of three elements – the tab, the tabstrip and the tab page – that combine to make the complete C1TabControl control.



The topics below describe each element of the C1TabControl control.

Tabs

Each tab on the C1TabControl control is represented by the C1TabItem class. The header of the tab is exposed by the Header property of the C1TabItem. Each tab is associated with a tab page (see [Tab Page](#) (page 13)).

Tabs can appear rounded, sloped, or rectangular. When a tab is added to a page, its default appearance is sloped – it looks similar to a tab on an office folder.

You can add a close button to each tab by setting the `TabItemClose` property to **InEachTab**. This will allow users to close any tab in the control; however, you can deny users the ability to close a particular tab by setting that tab's `CanUserClose` property to **False**.

Customizing the Header

When you want to add something simple to the tab's header, such as an unformatted string, you can simply use the common XML attributes in your XAML markup, such as in the following:

```
<c1:C1TabItem Header="Hello World"/>
```

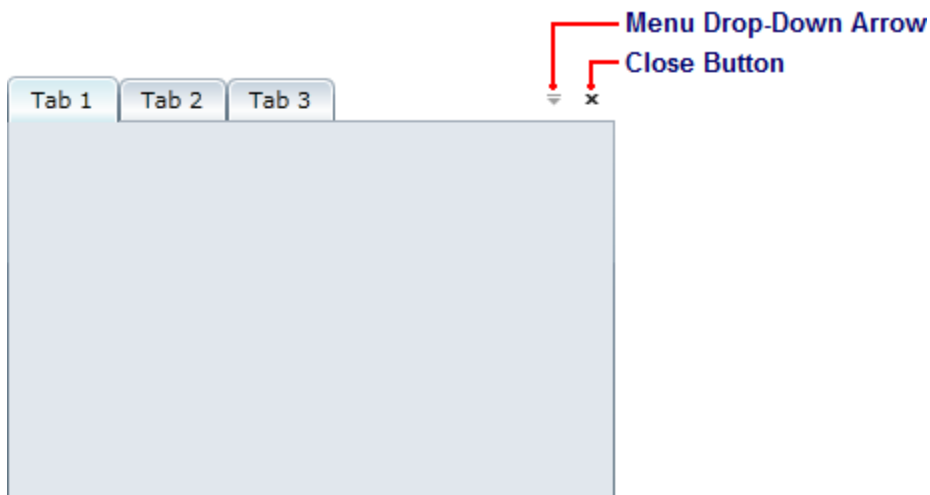
You can also customize the header using the `HeaderTemplate`.

Tabstrip

The `C1TabControl` control's tabstrip is created by adding one or more `C1TabItem` items to the control. Each tab in the strip is associated with a tab page (see [Tabs](#) (page 12) and [Tab Page](#) (page 13)).

By default, the tabstrip appears along the top of the `C1TabControl` control, but you can adjust the position of the strip by setting the `TabStripPlacement` property. You can also adjust the overlap of the tabs by setting the `TabStripOverlap` and `TabStripOverlapDirection` properties.

The tabstrip can also contain two optional items: a close button and a menu drop-down arrow. The close button, when clicked, closes the selected tab; the menu drop-down arrow, when clicked, exposes a drop-down menu from which users can open a different tab.



To learn more about the close button, see [Tab Closing](#) (page 16). To learn more about the menu drop-down, see [Optional Tab Menu](#) (page 16).

Tab Page

When a tab (`C1TabItem`) is added to a `C1TabControl` control, it will have a corresponding tab page that will initially appear as an empty space. In the tab page, you can add grids, text, images, and arbitrary controls. When working in Blend or Visual Studio's Design view, you can add elements to the tab page using a simple drag-and-drop operation.

You can add text to the tab page by setting the item's **Content** property or by adding a **TextBox** element to the tab page. Adding elements to the tab page at run time is simple: You can either use simple drag-and-drop operations or

XAML in Visual Studio or Blend. If you'd prefer to add a control at run time, you can use C# or Visual Basic code.

A `C1TabItem` item can only accept one child element at a time. However, you can circumvent this issue by adding a panel-based control as its child element. Panel-based controls, such as a **StackPanel** control, are able to hold multiple elements. The panel-based control meets the one control limitation of the `C1TabItem` item, but its ability to hold multiple elements will allow you to show several controls in the tab page.



Attribute Syntax versus Property Element Syntax

When you want to add something simple to the tab page, such as an unformatted string or a single control, you can simply use the common XML attributes in your XAML markup, such as in the following:

```
<c1:C1TabItem Content="Hello World"/>
```

However, there may be times where you want to add more complex elements, such as grids or panels, to the tab page. In this case you can use property element syntax, such as in the following:

```
<c1:C1TabItem>
    <c1:C1TabItem.Content>
        <StackPanel>
            <TextBlock Text="Hello"/>
            <TextBlock Text="World"/>
        </StackPanel>
    </c1:C1TabItem.Content>
</c1:C1TabItem>
```

You can also customize the content using the **ContentTemplate**.

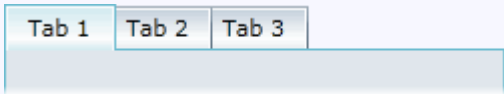
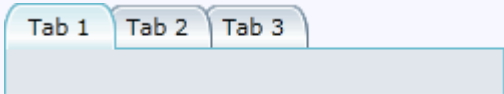
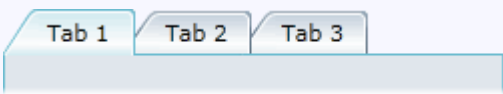
C1TabControl Features

This section details several important features of the `C1TabControl` control.

Tab Shaping

Tabs can appear rounded, sloped, or rectangular. By default, the tab will appear rounded, but you can change the shape of the tabs to any of the three settings by setting the `C1TabControl.TabItemShape` property to **Rectangle**, **Rounded**, or **Sloped**.

The following table illustrates each tab shape.


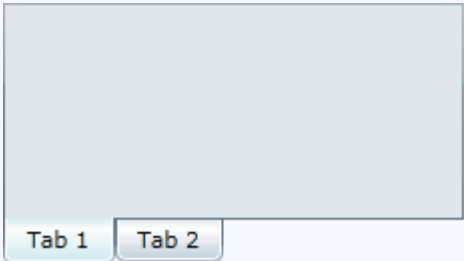
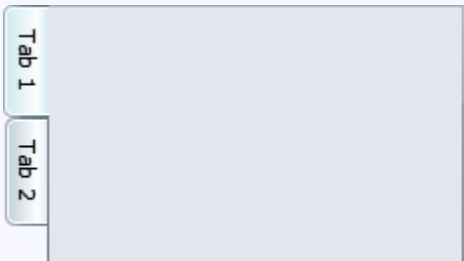
Tab Shape	Illustration
Rectangle	
Rounded	
Sloped	

For task-based help, see [Changing the Shape of Tabs](#) (page 31).

Tabstrip Placement

The **C1TabControl** control's tabstrip, by default, will appear along the top of the control. However, you can set the **C1TabControl.TabStripPlacement** property to **Bottom**, **Left**, or **Right** to change the position of the tabstrip.

The following table illustrates each **C1TabControl.TabStripPlacement** setting.

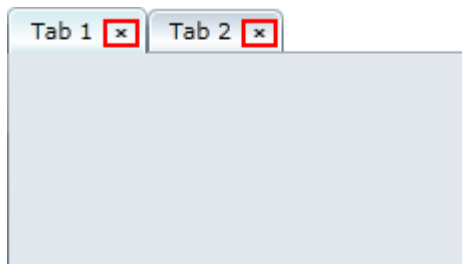
ExpandDirection	Result
Top	
Bottom	
Left	



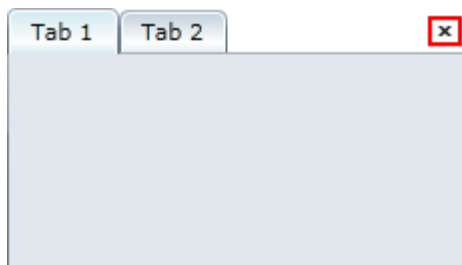
For task-based help, see [Changing the Tabstrip placement](#) (page 30).

Tab Closing

You can add a close button to each tab by setting the `TabItemClose` property to **InEachTab**. This will allow users to close any tab in the control. On-tab close buttons appear as follows:



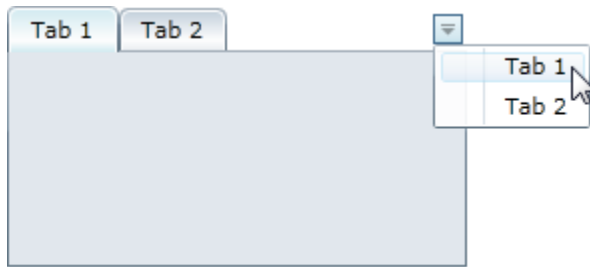
If you prefer, you can create a global close button that will appear on the tabstrip instead of directly on the tabs. To add the global close button, you set the `TabItemClose` property to **GlobalClose**. Clicking the global close button will close the currently selected tab. The global close button appears as follows:



If you want to prevent a user from closing a particular tab, just set that tab's `CanUserClose` property to **False**. For task-based help about allowing tab closing, see [Allowing Users to Close Tabs](#) (page 32); for task-based help about preventing tab closure, see [Preventing a User from Closing a Specific Tab](#) (page 33).

Optional Tab Menu

The `C1TabControl` control allows you to add a drop-down menu that allows users to select a tab and tab page from a menu. When enabled, this drop-down menu is accessible from the control's tabstrip. The drop-down menu appears as follows:



To activate the drop-down menu, simply set the **C1TabControl.TabStripMenuVisibility** property to **Visible**. For task-based help about adding a tab menu, see [Adding a Menu to the Tabstrip](#) (page 33).

Tab Overlapping

You can control the overlap of tabs by setting the **TabStripOverlap** property and the **TabStripOverlapDirection** property (for task-based help, see [Overlapping Tabs on a Tabstrip](#) (page 34)). The **TabStripOverlap** property controls how many pixels of the tab overlap, and the **TabStripOverlapDirection** direction property allows you to select an enumeration value that sets the direction of the overlap. The **TabStripOverlapDirection** property has three enumeration values – **Right**, **Left**, and **RightLeftFromSelected** – which are described and illustrated in the table below. Please note that the **TabStripOverlap** property for each of the following examples has been set to a value of 10.

Enumeration Value	Description	Illustration
Right	The rightmost tabs are in the back while the selected tab is in the front.	
Left	The leftmost tabs are in the back while the selected tab is in the front.	
RightLeftFromSelected	Leftmost tabs are in the back, rightmost tabs are in the back, and the selected tab is in the front.	

TabControl for WPF Layout and Appearance

The following topics detail how to customize the **C1TabControl** control's layout and appearance. You can use built-in layout options to lay your controls out in panels such as Grids or Canvases. Themes allow you to

customize the appearance of the grid and take advantage of WPF's XAML-based styling. You can also use templates to format and layout the control and to customize the control's actions.

ComponentOne ClearStyle Technology

ComponentOne ClearStyle™ technology is a new, quick and easy approach to providing Silverlight and WPF control styling. ClearStyle allows you to create a custom style for a control without having to deal with the hassle of XAML templates and style resources.

Currently, to add a theme to all standard WPF controls, you must create a style resource template. In Microsoft Visual Studio, this process can be difficult; this is why Microsoft introduced Expression Blend to make the task a bit easier. Having to jump between two environments can be a bit challenging to developers who are not familiar with Blend or do not have the time to learn it. You could hire a designer, but that can complicate things when your designer and your developers are sharing XAML files.

That's where ClearStyle comes in. With ClearStyle the styling capabilities are brought to you in Visual Studio in the most intuitive manner possible. In most situations you just want to make simple styling changes to the controls in your application so this process should be simple. For example, if you just want to change the row color of your data grid this should be as simple as setting one property. You shouldn't have to create a full and complicated-looking template just to simply change a few colors.

How ClearStyle Works

Each key piece of the control's style is surfaced as a simple color property. This leads to a unique set of style properties for each control. For example, a **Gauge** has **PointerFill** and **PointerStroke** properties, whereas a **DataGrid** has **SelectedBrush** and **MouseOverBrush** for rows.

Let's say you have a control on your form that does not support ClearStyle. You can take the XAML resource created by ClearStyle and use it to help mold other controls on your form to match (such as grabbing exact colors). Or let's say you'd like to override part of a style set with ClearStyle (such as your own custom scrollbar). This is also possible because ClearStyle can be extended and you can override the style where desired.

ClearStyle is intended to be a solution to quick and easy style modification but you're still free to do it the old fashioned way with ComponentOne's controls to get the exact style needed. ClearStyle does not interfere with those less common situations where a full custom design is required.

C1TabControl ClearStyle Properties

TabControl for WPF supports ComponentOne's new ClearStyle technology that allows you to easily change control colors without having to change control templates. By just setting a few color properties you can quickly style the entire grid.

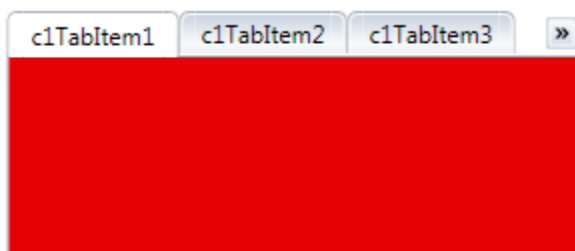
The following table outlines the brush properties of the **C1TabControl** control:

Brush	Description
Background	Gets or sets the brush for the background of each C1TabItem 's content area.
TabStripBackground	Gets or sets the brush for the background of the C1TabStrip 's background.
MouseOverBrush	Gets or sets the System.Windows.Media.Brush used to highlight the tabs when the mouse is hovered over them.
PressedBrush	Gets or sets the System.Windows.Media.Brush used to highlight the tabs when they are clicked on.

The following table outlines the brush properties of the **C1TabItem** control:

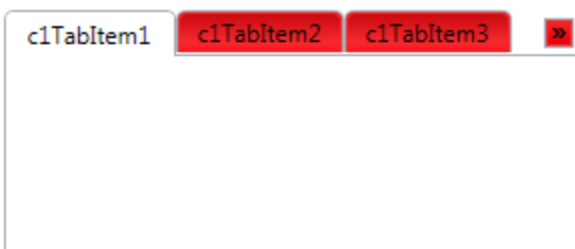
Brush	Description
Background	Gets or sets the backgrounds of all C1TabItems associated with the control.
MouseOverBrush	Gets or sets the System.Windows.Media.Brush used to highlight the tab when the mouse is hovered over them.
PressedBrush	Gets or sets the System.Windows.Media.Brush used to highlight the tab when they are clicked on.

You can completely change the appearance of the **C1TabControl** control by setting a few properties, such as the **Background** property, which sets the background color of the tab control's content area. For example, if you set the **Background** property to "#FFE40005", the **C1TabControl** control would appear similar to the following:



If you clicked through the tabs, you'd notice that the background of each tab's content area is red. If you'd like to change the color of one tab, simply set the **C1TabItem**'s **Background** property rather than the **C1TabControl**'s **Background** property.

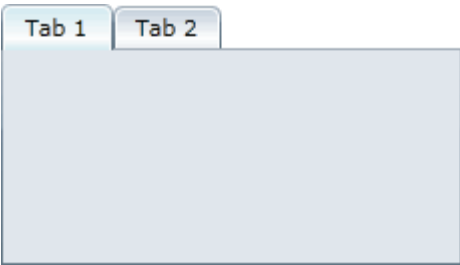
You can also set the color of the tabstrip by setting the **C1TabControl**'s **TabStripBackground** property. In the following example, the **TabStripBackground** property is set to "#FFE40005":



It's that simple with ComponentOne's ClearStyle technology. For more information on ClearStyle, see the [ComponentOne ClearStyle Technology](#) (page 18) topic.

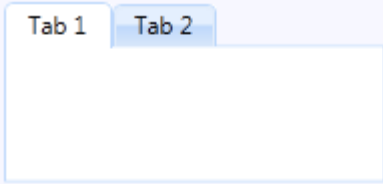
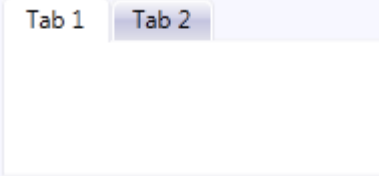

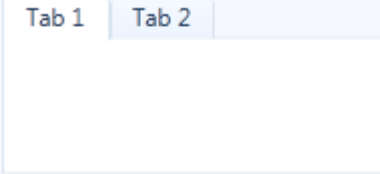

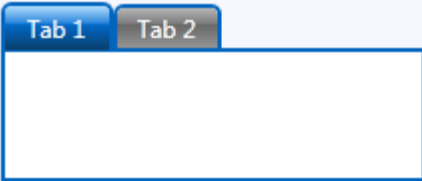

C1TabControl Themes

ComponentOne TabControl for WPF incorporates several themes that allow you to customize the appearance of your grid. When you first add a **C1TabControl** control with a **C1TabItem** to the page, it appears similar to the following image:



This is the control's default appearance. You can change this appearance by using one of the built-in themes or by creating your own custom theme. All of the built-in themes are based on WPF Toolkit themes. The built-in themes are described and pictured below.

Theme Name	Theme Preview
C1ThemeBureauBlack	
C1ThemeExpressionDark	
C1ThemeExpressionLight	
C1Blue	
C1ThemeOffice2007Black	

C1ThemeOffice2007Blue	 A UI element with two tabs, 'Tab 1' and 'Tab 2'. 'Tab 2' is selected and highlighted in a light blue color. The background is a solid light blue.
C1ThemeOffice2007Silver	 A UI element with two tabs, 'Tab 1' and 'Tab 2'. 'Tab 2' is selected and highlighted in a light grey color. The background is a solid light grey.
C1ThemeOffice2010Black	 A UI element with two tabs, 'Tab 1' and 'Tab 2'. 'Tab 2' is selected and highlighted in a dark grey color. The background is a solid dark grey.
C1ThemeOffice2010Blue	 A UI element with two tabs, 'Tab 1' and 'Tab 2'. 'Tab 2' is selected and highlighted in a light blue color. The background is a solid light blue.
C1ThemeOffice2010Silver	 A UI element with two tabs, 'Tab 1' and 'Tab 2'. 'Tab 2' is selected and highlighted in a light grey color. The background is a solid light grey.
C1ThemeShinyBlue	 A UI element with two tabs, 'Tab 1' and 'Tab 2'. 'Tab 2' is selected and highlighted in a dark blue color. The background is a solid dark blue.
C1ThemeWhistlerBlue	 A UI element with two tabs, 'Tab 1' and 'Tab 2'. 'Tab 2' is selected and highlighted in a light blue color. The background is a solid light blue.

To set an element's theme, use the **ApplyTheme** method. First add a reference to the theme assembly to your project, and then set the theme in code, like this:

- Visual Basic

```
Private Sub Window_Loaded(sender As System.Object, e As
System.Windows.RoutedEventArgs) Handles MyBase.Loaded
    Dim theme As New C1ThemeExpressionDark

    ' Using ApplyTheme
    C1Theme.ApplyTheme(LayoutRoot, theme)
```

- C#

```
private void Window_Loaded(object sender, RoutedEventArgs e)
{
    C1ThemeExpressionDark theme = new C1ThemeExpressionDark();

    //Using ApplyTheme
    C1Theme.ApplyTheme(LayoutRoot, theme);
}
```

To apply a theme to the entire application, use the **System.Windows.ResourceDictionary.MergedDictionaries** property. First add a reference to the theme assembly to your project, and then set the theme in code, like this:

- Visual Basic

```
Private Sub Window_Loaded(sender As System.Object, e As
System.Windows.RoutedEventArgs) Handles MyBase.Loaded
    Dim theme As New C1ThemeExpressionDark

    ' Using Merged Dictionaries
    Application.Current.Resources.MergedDictionaries.Add(C1Theme.GetCurrentThemeResources(theme))

End Sub
```

- C#

```
private void Window_Loaded(object sender, RoutedEventArgs e)
{
    C1ThemeExpressionDark theme = new C1ThemeExpressionDark();

    //Using Merged Dictionaries
    Application.Current.Resources.MergedDictionaries.Add(C1Theme.GetCurrentThemeResources(theme));
}
```

Note that this method works only when you apply a theme for the first time. If you want to switch to another ComponentOne theme, first remove the previous theme from **Application.Current.Resources.MergedDictionaries**.

TabControl for WPF Appearance Properties

ComponentOne TabControl for WPF includes several properties that allow you to customize the appearance of the control. You can change the appearance of the text displayed in the control and customize graphic elements of the control. The following topics describe some of these appearance properties.

Text Properties

The following properties let you customize the appearance of text in the C1TabControl control and its items.

Property	Description
----------	-------------

FontFamily	Gets or sets the font family of the control. This is a dependency property.
FontSize	Gets or sets the font size. This is a dependency property.
FontStretch	Gets or sets the degree to which a font is condensed or expanded on the screen. This is a dependency property.
FontStyle	Gets or sets the font style. This is a dependency property.
FontWeight	Gets or sets the weight or thickness of the specified font. This is a dependency property.
TextAlignment	Gets or sets how the text should be aligned in the tab. This is a dependency property.
Header	Gets or sets the header of a tab item.
HeaderFontFamily	Gets or sets the font family of the header.
HeaderFontStretch	Gets or sets the font stretch of the header.
HeaderFontStyle	Gets or sets the font style of the header.
HeaderFontWeight	Gets or sets the font weight of the header.

Content Positioning Properties

The following properties let you customize the position of header and content area content in the C1TabControl control and its items.

Property	Description
HeaderPadding	Gets or sets the padding of the header.
HeaderHorizontalContentAlignment	HorizontalContentAlignment of the header.
HeaderVerticalContentAlignment	Gets or sets the vertical content alignment of the header.
HorizontalContentAlignment	Gets or sets the horizontal alignment of the control's content. This is a dependency property.
VerticalContentAlignment	Gets or sets the vertical alignment of the control's content. This is a dependency property.

Color Properties

The following properties let you customize the colors used in the C1TabControl control and its items.

Property	Description
Background	Gets or sets a brush that describes the background of a control. This is a dependency property.
Foreground	Gets or sets a brush that describes the foreground color. This is a dependency property.
HeaderBackground	Gets or sets the background brush of the header.
HeaderForeground	Gets or sets the foreground brush of the header.

Border Properties

The following properties let you customize the border of the C1TabControl control and its items.

Property	Description
BorderBrush	Gets or sets a brush that describes the border background of a control. This is a dependency property.
BorderThickness	Gets or sets the border thickness of a control. This is a dependency property.

Size Properties

The following properties let you customize the size of the C1TabControl control and its items.

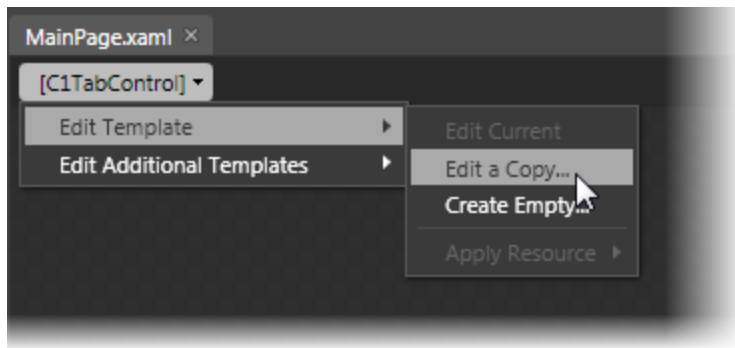
Property	Description
Height	Gets or sets the suggested height of the element. This is a dependency property.
MaxHeight	Gets or sets the maximum height constraint of the element. This is a dependency property.
MaxWidth	Gets or sets the maximum width constraint of the element. This is a dependency property.
MinHeight	Gets or sets the minimum height constraint of the element. This is a dependency property.
MinWidth	Gets or sets the minimum width constraint of the element. This is a dependency property.
Width	Gets or sets the width of the element. This is a dependency property.

TabControl Templates

One of the main advantages to using a WPF control is that controls are "lookless" with a fully customizable user interface. Just as you design your own user interface (UI), or look and feel, for WPF applications, you can provide your own UI for data managed by **ComponentOne TabControl for WPF**. Extensible Application Markup Language (XAML; pronounced "Zammel"), an XML-based declarative language, offers a simple approach to designing your UI without having to write code.

Accessing Templates

You can access templates in Microsoft Expression Blend by selecting the C1TabControl control and, in the menu, selecting **Edit Template**. Select **Edit a Copy** to create an editable copy of the current template or select **Create Empty** to create a new blank template.



If you want to edit the C1TabItem template, simply select the C1TabItem control and, in the menu, select **Edit Template**. Select **Edit a Copy** to create an editable copy of the current template or **Create Empty**, to create a new blank template.

Note: If you create a new template through the menu, the template will automatically be linked to that template's property. If you manually create a template in XAML you will have to link the appropriate template property to the template you've created.

Note that you can use the [Template](#) property to customize the template.

Additional Templates

In addition templates, the C1TabControl control and C1TabItem control include a few additional templates. These additional templates can also be accessed in Microsoft Expression Blend – in Blend select the C1TabControl or C1TabItem control and, in the menu, select **Edit Additional Templates**. Choose a template, and select **Create Empty**.

Item Templates

ComponentOne TabControl for WPF's tab control is an **ItemsControls** that serves as a container for other elements. As such, the control includes templates to customize items places within the tab control. These templates include an **ItemTemplate**, an **ItemsPanel**, and an **ItemContainerStyle** template. You use the **ItemTemplate** to specify the visualization of the data objects, the **ItemsPanel** to define the panel that controls the layout of items, and the **ItemStyleContainer** to set the style of all container items.

Accessing Templates

You can access these templates in Microsoft Expression Blend by selecting the C1TabControl control and, in the menu, selecting **Edit Additional Templates**. Choose **Edit Generated Items (ItemTemplate)**, **Edit Layout of Items (ItemsPanel)**, or **Edit Generated Item Container (ItemStyleContainer)** and select **Create Empty** to create a new blank template or **Edit a Copy**.

A dialog box will appear allowing you to name the template and determine where to define the template.

TabControl for WPF Samples

Please be advised that these ComponentOne software tools are accompanied by various sample projects and/or demos which may make use of other development tools included with the ComponentOne Studios.

Samples can be accessed from the **ComponentOne Control Explorer**. To view samples, on your desktop, click the **Start** button and then click **ComponentOne | Studio for WPF | Samples | WPF ControlExplorer**.

The following pages within the **ControlExplorer** detail the C1TabControl control:

Sample	Description
TabControl	Illustrates the functionality of the C1TabControl control.

TabControl for WPF Task-Based Help

The task-based help assumes that you are familiar with programming in Visual Studio .NET and know how to use the C1TabControl control in general. If you are unfamiliar with the **ComponentOne TabControl for WPF** product, please see the **TabControl for WPF** Quick Start first.

Each topic in this section provides a solution for specific tasks using the **ComponentOne TabControl for WPF** product.

Each task-based help topic also assumes that you have created a new WPF project.

Adding a Tab to the C1TabControl Control

In this topic, you will add tabs to the C1TabControl control in Design view, in XAML, and in code.

In Design View in Blend

Complete the following steps:

1. Click the C1TabControl control once to select it.
2. In the Toolbox, double-click the C1TabItem icon to add a tab item to the C1TabControl control.

A C1TabItem appears within the C1TabControl control.

In XAML

To add a tab to the C1TabControl control, place the following markup between the `<c1:C1TabControl>` and `</c1:C1TabControl>` tags:

```
<c1:C1TabItem Content="c1TabItem">
</c1:C1TabItem>
```

In Code

Complete the following steps:

1. Open the **Window1.xaml.cs** page.
2. Import the following namespace:
 - Visual Basic
`Imports C1.WPF`
 - C#
`using C1.WPF;`
3. Add the following code beneath the **InitializeComponent()** method:

- Visual Basic

```
'Create the tab and add content
Dim C1TabItem1 As New C1TabItem()
C1TabItem1.Content = "C1TabItem1"

'Add the tab to the control
c1TabControl1.Items.Add(C1TabItem1)
```

- C#

```
//Create the tab and add content
c1TabItem c1TabItem1 = new c1TabItem();
c1TabItem1.Content = "c1TabItem1";
//Add the tab to the control
c1TabControl1.Items.Add(c1TabItem1);
```

4. Run the program.

Adding Content to a Tab Page

You can add content to the tab page using the **Content** property. This topic demonstrates how to add content – in this case, a standard **Button** control – to a tab page in Design view, in XAML, and in code. This topic assumes that you have added at least one **C1TabItem** to the **C1TabControl** control.

In Design View

Complete the following steps:

1. In Design view, select the tab you wish to add the content to.
2. In the Toolbox, double-click the **Button** icon to add a **Button** control to the tab.
3. Double click the **Button** icon to add it to the tab page's content area.
4. Select the Button control and set the following properties in the **Properties** window:
 - Set the **Width** property to "Auto".
 - Set the **Height** property to "Auto".
5. Run the program and click the tab.

In XAML

To add a **Button** control to the content area in XAML, place the following markup between the `<c1:C1TabItem>` and `</c1:C1TabItem>` tags:

```
<Button Content="Button" Height="Auto" Width="Auto"/>
```

In Code

Complete the following steps:

1. Open the **Window1.xaml.cs** page.
2. Add the following code beneath the **InitializeComponent()** method:
 - Visual Basic

```
'Create the Button control
Dim NewButton As New Button()
NewButton.Content = "Button"

'Set the Button Control's Width and Height properties
NewButton.Width = Double.NaN
NewButton.Height = Double.NaN

'Add the Button to the content area
c1TabItem1.Content = (NewButton)
```

- C#


```
//Create the Button control
Button NewButton = new Button();
NewButton.Content = "Button";

//Set the Button Control's Width and Height properties
NewButton.Width = double.NaN;
NewButton.Height = double.NaN;

//Add the Button to the content area
c1TabItem1.Content = (NewButton);
```

3. Run the program.

✓ This Topic Illustrates the Following:

The image below depicts a `C1TabItem` with a **Button** control as content.



Specifying a Tab Header

In this topic, you will add a header to a tab by setting the **Header** property Blend, in XAML, and in code. This topic assumes that you have added at least one `C1TabItem` to the `C1TabControl` control.

In XAML

To add a header to a tab, add `Header="Hello World"` to the `<c1:C1TabItem>` tag so that the markup resembles the following:

```
<c1:C1TabItem Header="Hello World"></c1:C1TabItem>
```

In Code

Complete the following steps:

1. Open the **Window1.xaml.cs** page.
2. Add the following code beneath the **InitializeComponent()** method:
 - Visual Basic

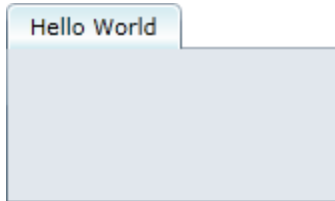

```
C1TabItem1.Header = Hello World
```
 - C#

```
c1TabItem1.Header = Hello World;
```

3. Run the program.

✔ **This Topic Illustrates the Following:**

The image below depicts a C1TabItem with a header.



Changing the Tabstrip Placement

The tabstrip of a C1TabControl control is placed on the top by default, but you can also place it to the left, right, or bottom of the control by setting the TabStripPlacement property. In this topic, you will set the TabStripPlacement property to **Right** in Design view, in XAML, and in code. For more information, see [Tabstrip Placement](#) (page 15). For more information on tabstrip placement settings, see [Tabstrip Placement](#) (page 15).

In Design View

Complete the following steps:

1. Select the C1TabControl control.
2. In the **Properties** window, click the TabStripPlacement drop-down arrow and select **Right** from the list.

In XAML

To change the tabstrip placement, add `TabStripPlacement="Right"` to the `<c1:C1TabControl>` tab so that the markup resembles the following:

```
<c1:C1TabControl TabStripPlacement="Right"></c1:C1TabControl>
```

In Code

Complete the following steps:

1. Open the **Window1.xaml.cs** page.
2. Add the following code beneath the **InitializeComponent()** method:
 - Visual Basic

```
C1TabControl1.TabStripPlacement = Right
```
 - C#

```
c1TabControl1.TabStripPlacement = Right;
```
3. Run the program.

✔ **This Topic Illustrates the Following:**

The image below depicts a C1TabControl control with its tabstrip placed on its right side.



Changing the Shape of Tabs

The tabs of a C1TabControl control can be rectangular, rounded, or sloped. By default, the tabs are rounded, but you can change the shape of the tabs by setting the C1TabControl control's TabItemShape property. In this topic, you will change the shape of the tabs to **Sloped** in Design view, in XAML, and in code. For more information on tabstrip shaping settings, see [Tab Shaping](#) (page 14).

In Design View

Complete the following steps:

1. Select the C1TabControl control.
2. In the **Properties** window, click the TabItemShape drop-down arrow and select **Sloped** from the list.

In XAML

To change the shape of the tabs, add `TabItemShape="Sloped"` to the `<c1:C1TabControl>` tag so that the markup resembles the following:

```
<c1:C1TabControl TabItemShape="Sloped"></c1:C1TabControl>
```

In Code

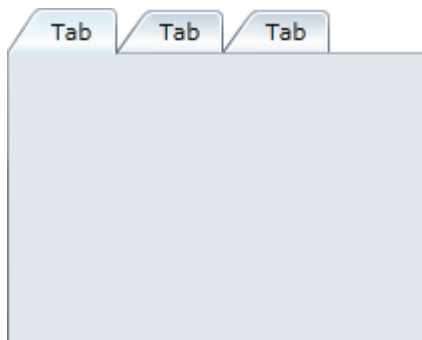
Complete the following steps:

1. Open the **Window1.xaml.cs** page.
2. Add the following code beneath the **InitializeComponent()** method:
 - Visual Basic
`C1TabControl1.TabItemShape = Sloped`
 - C#
`c1TabControl1.TabItemShape = Sloped;`
3. Run the program.



This Topic Illustrates the Following:

The image below depicts a C1TabControl control with sloped tabs.



Allowing Users to Close Tabs

By default, users can't close the tabs on a C1TabControl control. You can alter this by setting the `TabItemClose` property to **InEachTab** or **GlobalClose** so that users can close tabs by clicking a button inside the tab or by selecting a tab and then clicking a universal close button (see [Tab Closing](#) (page 16) for more information). In this topic, you will set the `TabItemClose` property to **GlobalClose** in Design view, in XAML, and in code.

In Design View

Complete the following steps:

1. Select the C1TabControl control.
2. In the **Properties** window, click the `TabItemClose` drop-down arrow and select **GlobalClose** from the list.

In XAML

To allow users to close tabs using a universal button, add `TabItemClose="GlobalClose"` to `<c1:C1TabControl>` tab so that the markup resembles the following:

```
<c1:C1TabControl TabItemClose="GlobalClose"></c1:C1TabControl>
```

In Code

Complete the following steps:

1. Open the **Window1.xaml.cs** page.
2. Enter Code view and add the following code beneath the **InitializeComponent()** method:
 - Visual Basic
`C1TabControl1.TabItemClose = GlobalClose`
 - C#
`c1TabControl1.TabItemClose = GlobalClose;`
3. Run the program.



This Topic Illustrates the Following:

The image below depicts a C1TabControl control tabstrip with a global close button.



Preventing a User from Closing a Specific Tab

When tab closing enabled (see [Tab Closing](#) (page 16) and [Allowing Users to Close Tabs](#) (page 32) for more information), users can disable any tab on the strip by default. However, you can prevent users from closing a specific tab by setting that `C1TabItem` item's `CanUserClose` property to **False**.

In Design View

Complete the following steps:

1. Select the tab that you wish to prevent users from closing.
2. In the **Properties** window, clear the `CanUserClose` check box.

In XAML

To prevent a user from closing a tab, add `CanUserClose="False"` to the `<c1:C1TabItem>` tag of the tab you want to prevent users from closing. The XAML will resemble the following:

```
<c1:C1TabItem CanUserClose="False"></c1:C1TabItem>
```

In Code

Complete the following steps:

1. Open the **Window1.xaml.cs** page.
2. Add the following code beneath the **InitializeComponent()** method:
 - Visual Basic

```
C1TabItem1.CanUserClose = False
```
 - C#

```
c1TabItem1.CanUserClose = false;
```
3. Run the program.

Adding a Menu to the Tabstrip

You can add a menu to the tabstrip so that users can open and close tabs from a context menu instead of by clicking on tabs. To add the tab menu, set the `C1TabControl` control's `TabStripMenuVisibility` property to **Visible**.

In Design View

Complete the following steps:

1. Select the `C1TabControl` control.
2. In the **Properties** window, click the `TabStripMenuVisibility` drop-down arrow and select **Visible** from the list.

In XAML

To add a menu to the tabstrip, add `TabStripMenuVisibility="Visible"` to the `<c1:C1TabControl>` tag so that the markup resembles the following:

```
<c1:C1TabControl TabStripMenuVisibility="Visible"></c1:C1TabControl>
```

In Code

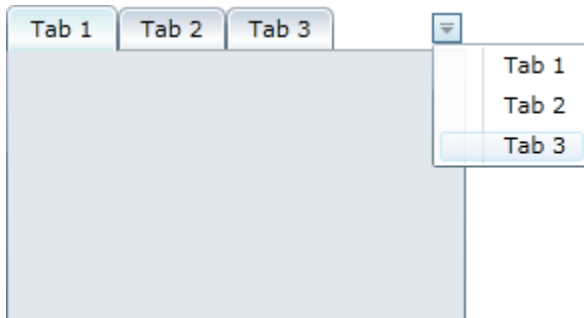
Complete the following steps:

1. Open the **Window1.xaml.cs** page.
2. Add the following code beneath the **InitializeComponent()** method:
 - Visual Basic
`C1TabControl1.TabStripMenuVisibility = Visible`
 - C#
`c1TabControl1.TabStripMenuVisibility = Visible;`
3. Run the program.



This Topic Illustrates the Following:

The image below depicts a C1TabControl control tabstrip with tab menu.



Overlapping Tabs on a Tabstrip

You can control the overlapping of tabs by setting the `TabStripOverlap` property and the `TabStripOverlapDirection` property (for more information, see [Tab Overlapping](#) (page 17)). In this topic, you'll set the `TabStripOverlap` property and `TabStripOverlapDirection` properties in design view, in XAML, and in code. This topic assumes that you have added a C1TabControl with at least three tabs to your project (see [Adding a Tab to the C1TabControl Control](#) (page 27)).

In Design View

Complete the following steps:

1. Select the C1TabControl control.
2. In the Properties window, set the following properties:
 - Set the `TabStripOverlap` property to "12".
 - Set the `TabStripOverlapDirection` property to **Left**.

In XAML

Add `TabStripOverlap="12"` and `TabStripOverlapDirection="Left"` to the `<c1:C1TabControl>` tab so that the markup resembles the following:

```
<c1:C1TabControl HorizontalAlignment="Left" VerticalAlignment="Top"
Height="156" Width="226" TabStripOverlap="12"
TabStripOverlapDirection="Left">
```

In Code

Complete the following steps:

1. Enter Code view and add the following code beneath the **InitializeComponent()** method:

- Visual Basic

```
C1TabControl1.TabStripOverlap = 12
C1TabControl1.TabStripOverlapDirection = C1.Silverlight.
C1TabPanelOverlapDirection.Left
```

- C#

```
c1TabControl1.TabStripOverlap = 12;
C1TabControl1.TabStripOverlapDirection = C1.Silverlight.
c1TabPanelOverlapDirection.Left;
```

2. Run the program.



This Topic Illustrates the Following:

The following image depicts the result of this topic.

