ComponentOne

Sparkline for UWP

ComponentOne, a division of GrapeCity
201 South Highland Avenue, Third Floor
Pittsburgh, PA 15206 USA

Website: http://www.componentone.com
Sales: 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 media on which the software is delivered is 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 media 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 the defective media by sending it and a check for $2 5 (to cover postage and handling) to ComponentOne.

Except for the express warranty of the original media on which the software is delivered is 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. ComponentOne is 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.

Table of Contents

Copyright © 2017 GrapeCity, inc. All rights reserved.

Sparkline for UWP

Get a light charting control for simple data visualization with **Sparkline for UWP**. Sparklines are a great way to visualize trends in a small space, such as in data templates and on dashboard tiles.

Help with UWP Edition

Getting Started

For information on installing **ComponentOne Studio UWP Edition**, licensing, technical support, namespaces and creating a project with the control, please visit Getting Started with ComponentOne Studio UWP Edition.

Copyright © 2017 GrapeCity, inc. All rights reserved.

Key Features

Sparkline for UWP includes the following features:

- **Supports 3 Chart Types**

  The C1Sparkline class supports three different chart types: column, line, and win-loss.

- **Display Colors for Marker Points**

  The sparklines can display colors for the marker points. You can set colors for the high, low, negative, first, and last points.

- **Flexible Data-Binding**

  Bind the C1Sparkline to any enumerable collection of numeric data values. You can populate the sparklines in code or configure their bindings in XAML.

- **Supports a Date Axis**

  Not only does the C1Sparkline control support horizontal and vertical axes, but you can also configure a date axis by supplying a collection of dates that act as x-coordinates.

Copyright © 2017 GrapeCity, inc. All rights reserved.

Sparkline for UWP Quick Start

The following quick start guide is intended to get you up and running with **Sparkline for UWP**. In this quick start you'll start in Visual Studio and create a new project, add **Sparkline for UWP** controls to your application, and customize the appearance and behavior of the controls.

Step 1 of 3: Creating the Application

In this step you'll create a new Windows application in Visual Studio and add the C1Sparkline control to a page.

1. In Visual Studio, select **File | New | Project**.
2. In the **New Project** dialog box:

    2. Select Templates | Visual C# | Windows | Universal. From the templates list, select **Blank App (Universal Windows)**.
    3. Enter a **Name** and click **OK** to create your project.
3. Right-click the **References** folder in the Solution Explorer and select **Add Reference** from the dropdown list.
4. Expand Universal Windows and select Extensions; you should see the **UWP Assemblies** in the center pane.
5. Check the **C1.UWP** and **C1.UWP.Sparkline** references and click **OK**.
6. Add the following markup to the opening <Page> tag:

    Markup
    ```
    xmlns:c1="using:C1.Xaml.Sparkline"
    ```

    The <Page> tag should resemble the following:

    Markup
    ```
    <Page
    x:Class="App2.MainPage"
       xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
       xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
       xmlns:local="using:App2"
       xmlns:c1="using:C1.Xaml.Sparkline"
       xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
       xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
       mc:Ignorable="d">
    ```
7. Place your cursor between the <Grid> </Grid> tags.
8. Locate the C1Sparkline control in the Visual Studio ToolBox. Double-click the control to add it to your application.
9. Edit the <c1:C1Sparkline/> tag so that it resembles the following sample. This will add a name so that the control can be called in code, add some color customization to the sparkline control, and will add the appropriate binding statements:

    Markup
    ```
    <c1:C1Sparkline x:Name="sparkline" Width="100" Height="100" Data="{Binding
    Data}" DateAxisData="{Binding DateAxis}" SeriesColor="#FF4BC128"
    ShowMarkers="True"/>
    ```

What You've Accomplished

Copyright © 2017 GrapeCity, inc. All rights reserved.

In this step, you created a new Windows Store application, added the appropriate assembly references to the application, and added a C1Sparkline control.

Step 2 of 3: Adding Code

In this step, you will add code to specify the data to which the C1Sparkline control is bound.

1. Right-click MainPage.xaml and select **View Code** from the list.
2. Add the following code below the **InitializeComponent()** method to change the marker color, and to create the data for your C1Sparkline control:

```
C#
sparkline.MarkersColor = Colors.DeepPink;
List<double> data = new List<double>() { 1, -2, 3, 4};
            Sale = new Sales();
            Sale.Data = data;

            List<DateTime> dateTime = new List<DateTime>() { new
DateTime(2013,11,1), new DateTime(2013,11,2), new DateTime(2013,11,4), new
DateTime(2013,10,5) };
            Sale.DateAxis = dateTime;
            this.sparkline.DataContext = Sale;
```

3. Next, add the following class below the code added in point no. 2:

```
C#
public Sales Sale { get; set; }
```

4. Add the following code to populate the Sparkline with data:

```
C#
public class Sales
    {
        public List<double> Data{get;set;}
        public List<DateTime> DateAxis{get;set;}
        public Sales()
        {
            Data = new List<double>();
            DateAxis = new List<DateTime>();
        }
    }
```

What You've Accomplished:

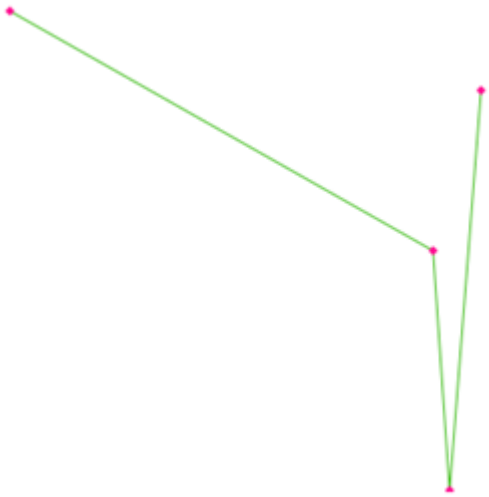
In this topic, you added code to bind the C1Sparkline control to a data source.

Step 3 of 3: Run Your Application

In this step, you will run your application

Copyright © 2017 GrapeCity, inc. All rights reserved.

Press **F5** or start debugging to run your application. It should resemble the following image:



Note that the SeriesColor (the line) is bright green and the markers are pink.

What You've Accomplished

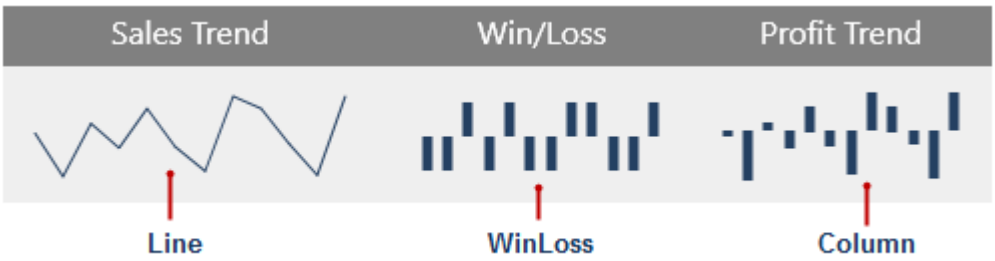
Congratulations! You've completed the **Sparkline for UWP** Quick Start! You created a sparkline application, added data to the C1Sparkline control, and customized the appearance of the chart.

Copyright © 2017 GrapeCity, inc. All rights reserved.

Sparkline for UWP Concepts and Main Properties

The following topics cover basic aspects of the **Sparkline for UWP** control's properties. These properties allow you to customize your C1Sparkline control, from the axes' and control's appearance to the data contained in the chart.

The C1Sparkline control allows you to create a succinct and unbiased data analysis on the same line as your data. Three different types of C1Sparkline give you the flexibility to represent graphically many different types of data. The image below shows the three types of C1Sparkline control:



The steps involved in creating a typical sparkline are:

1. **Choose the sparkline type (**C1Sparkline.SparklineType **property)**

   C1Sparkline supports 3 types: Column, Line, and WinLoss. The best type depends largely on the nature of the data, and will be discussed later.

2. **Set up the axes**

   Setting up the axes typically involves specifying the visibility and type of X Axis your wish to use, and the type, minimum value, and maximum value for the vertical axis.

2. **Add a data series (**C1Sparkline.Data**)**

   This step involves creating the data and binding to it, or binding to an outside data source.

4. **Adjust the chart's appearance using the Appearance properties.**

Axes

The following properties below represent the axes in C1Sparkline:

Property	Description	Code Sample
DisplayDateAxis	When set to true, this property displays data over a span of dates on the X Axis. This type of axis allows gaps in data if there are no data points for a particular date.	C1Sparkline.DisplayDateAxis = true
DisplayXAxis	This property can be set to "true" or "false" to indicate whether to display the X Axis.	C1Sparkline.DisplayXAxis = true
ManualMax	Gets or sets the maximum value for the vertical axis that is shared across all sparklines in a sparkline group. The value must be zero if maxAxisType isn't set to	C1Sparkline.ManualMax = 20

Copyright © 2017 GrapeCity, inc. All rights reserved.

	"custom".	
ManualMin	Gets or sets the minimum value for the vertical axis that is shared across all sparklines in a sparkline group. The value must be zero if minAxisType isn't set to "custom".	C1Sparkline.ManualMin = -5
MaxAxisType	This property specifies how the vertical axis maximums for the sparklines in a sparkline group are calculated.	C1Sparkline.MaxAxisType = SparklineAxisMinMax.Group
MinAxisType	This property specifies how the vertical axis minimums for the sparklines in a sparkline group are calculated.	C1Sparkline.MinAxisType = SparklineAxisMinMax.Group

Data and Data Binding

There are two properties that control the data used in the C1Sparkline control: Data and DateAxisData. These properties are described in the table below:

Property	Description	Code Sample
Data	Provides C1Sparkline's values.	C1Sparkline.Data = new List<double>(){1,2,3,4,-1,-3,-4.5,6}
DateAxisData	Provides the values for C1Sparkline's DateTime Axis.	C1Sparkline.DateAxisData = new List<DateTime>(){ new DateTime(2013,11,1), new DateTime(2013,11,2)}

The steps required to create data bound charts are as follows:

1. Choose the sparkline type (SparklineType property).
2. Set the Binding property to the collection of items that contain the desired sparkline data. This can be seen in the Sparkline for UWP Quick Start.
3. Adjust the chart's appearance using the Appearance properties.

Sparkline Types

The C1Sparkline control allows you to create a small, inline chart to graphically represent your data.

To set up a Column sparkline, specify the corresponding string in the C1Sparkline.SparklineType property:

Markup

```
<C1:C1Sparkline x:Name="c1sparkline1" SparklineType="Column" >
    . . .
</C1:C1Sparkline>
```

The available C1Sparkline types are specified by the members of enumeration SparklineType.

Column Sparkline

Copyright © 2017 GrapeCity, inc. All rights reserved.

The Column C1Sparkline is useful for representing data where previous values and the current value don't closely affect one another. Types of non-continuous data like sports scores and cash register receipts are best represented by a Column C1Sparkline. The following image shows a C1Sparkline control with the SparklineType property set to Column.

Line Sparkline

The Line **C1Sparkline** is useful for representing a continuous flow of data, such as stock values or sales data. The following image shows a C1Sparkline control with the SparklineType property set to Line:

WinLoss Sparkline

A WinLoss type C1Sparkline control is useful for representing true/false or win/loss data. Unlike the Column type sparkline, the WinLoss columns are all the same size.

Sparkline Appearance

Using built-in color properties is the simplest way to customize your **Sparkline** control.

C1Sparkline Appearance Properties

The C1Sparkline control allows you to change its appearance using many different properties. You can determine the type of sparkline created, the visibility of the markers and the axis; the colors applied to the axis, markers, and series; and the direction of the data.

Two general appearance properties can be seen in the table below:

Property Name	Property Description	Code Sample
RightToLeft	Allows data to be displayed right-to-left. By default, data is displayed left-to-right.	C1Sparkline.RightToLeft = true
SparklineType	Determines the type of sparkline control.	C1Sparkline.SparklineType = SparklineType.Column

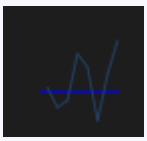
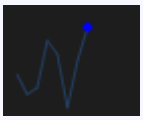
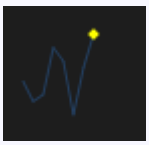
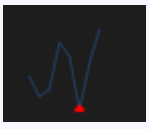
The visibility properties can be seen in the table below:

Property Name	Property Description	Code Sample

Copyright © 2017 GrapeCity, inc. All rights reserved.

ShowFirst	Determines the visibility of the first data marker	C1Sparkline.ShowFirst = true
ShowLast	Determines the visibility of the last data marker.	C1Sparkline.ShowLast = true
ShowLow	Determines the visibility of the lowest data marker.	C1.Sparkline.ShowLow = true
ShowHigh	Determines the visibility of the highest data marker.	C1Sparkline.ShowHigh = true
ShowMarkers	Determines the visibility of all data markers.	C1Sparkline.ShowMarkers = true
ShowNegative	Determines the visibility of the negative data markers.	C1Sparkline.ShowNegative = true
DisplayEmptyCellsAs	Determines how empty cells are displayed.	C1Sparkline.DisplayEmptyCellsAs= EmptyValueStyle.Gaps

The color properties can be seen in the table below:

Property Name	Property Description	Code Sample	Image
AxisColor	Gets or sets the x-axis color. Note that the **DisplayXAxis** property must be set to true.	AxisColor = Colors.Red	
FirstMarkerColor	Gets or sets the color of the first data point for each sparkline in the sparkline group. Note that the **ShowFirst** property must be set to true.	FirstMarkerColor = Colors.Red	
HighMarkerColor	Gets or sets the color for the highest data point for each sparkline in the sparkline group. Note that the **ShowHigh** property must be set to true.	HighMarkerColor = Colors.Blue	
LastMarkerColor	Gets or sets the color for the last data point for each sparkline in the sparkline group. Note that the **ShowLast** property must be set to true.	LastMarkerColor = Colors.Yellow	
LowMarkerColor	Gets or sets the color for the lowest data point for each sparkline in the sparkline group. Note that the **ShowLow** property must be set to true.	LowMarkerColor = Colors.Red	
MarkersColor	Specifies the color of the data markers	MarkersColor =	

Copyright © 2017 GrapeCity, inc. All rights reserved.

	for each sparkline in the sparkline group. Note that the **ShowMarkers** property must be set to true.	Colors.DeepPink	
NegativeColor	Specifies the color of the negative data points for each sparkline in the sparkline group. Note that the **ShowNegative** property must be set to true.	NegativeColor = Colors.Pink	
SeriesColor	Specifies the color for the line in the line chart. This property is only available for the line chart type C1Sparkline control.	SeriesColor = Colors.Brown	

Copyright © 2017 GrapeCity, inc. All rights reserved.

Sparkline for UWP Tutorials

The following tutorials assume that you are familiar with programming in Visual Studio. The tutorials provide step-by-step instructions; no prior knowledge of **Sparkline for UWP** is needed. By following the steps outlined in this section, you will be able to create projects demonstrating **Sparkline for UWP** features.

You are encouraged to run the tutorial projects, and experiment with your own modifications.

Adding C1Sparkline to a List Box

In this tutorial, you will create a C1Sparkline control that charts data in a ListBox control.

Step 1 of 4: Creating the Application

In this step, you'll create a new Universal Windows application in Visual Studio, add assembly references, and add the .xaml and code files needed for the application.

1. Select **File | New | Project** to open the **New Project** dialog box.
    1. Select Templates | Visual C# | Windows | Universal. From the templates list, select **Blank App (Universal Windows)**.
    2. Enter a name for your application, in this case RegionSales, and click OK. A new, blank Universal Windows application will open.

1. In the Solution Explorer, right-click the **References** file and select **Add Reference** from the list. Browse to locate the following assembly references:

- C1.UWP.dll
- C1.UWP.Sparkline.dll

1. Double-click the MainPage.xaml file to open it.
2. Add the following namespace declarations to the <Page> tag at the top of the page:

- xmlns:local="using:RegionSales"
- xmlns:sp="using:C1.Xaml.Sparkline"

The <Page> tag at the top of the page should resemble the following sample:

Markup

```
<Page
    x:Class="RegionSales.RegionSale"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:RegionSales"
    xmlns:sp="using:C1.Xaml.Sparkline"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d">
```

1. Place your cursor between the <Grid> </Grid> tags and insert the following markup. This will create the grid's resources, row, and column definitions:

Markup

Copyright © 2017 GrapeCity, inc. All rights reserved.

```
<Grid.Resources>
    <Style TargetType="TextBlock">
        Setter Property="FontSize" Value="16" />
    </Style>
</Grid.Resources>
<Grid Width="800" Margin="10">
    <Grid.RowDefinitions>
        <RowDefinition Height="30"/>
        <RowDefinition />
    </Grid.RowDefinitions>
    <Grid Background="Gray">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="100"/>
            <ColumnDefinition Width="100"/>
               <ColumnDefinition Width="100"/>
            <ColumnDefinition Width="200"/>
            <ColumnDefinition Width="150"/>
            <ColumnDefinition Width="150"/>
        </Grid.ColumnDefinitions>
    </Grid>
</Grid>
```

1. Directly after the </Grid.ColumnDefinitions> tag, add the following markup to create the label TextBox controls, and a separate ScrollViewer control which contains the **RegionSalesListBox**:

Markup

```
<TextBlock Text="Region" HorizontalAlignment="Center" VerticalAlignment="Center"/>
                <TextBlock Text="Total Sales" Grid.Column="1"
HorizontalAlignment="Center" VerticalAlignment="Center"/>
                <TextBlock Text="Net Sales" Grid.Column="2"
HorizontalAlignment="Center" VerticalAlignment="Center"/>
                <TextBlock Text="Sales Trend" Grid.Column="3"
HorizontalAlignment="Center" VerticalAlignment="Center"/>
                <TextBlock Text="Win/Loss" Grid.Column="4"
HorizontalAlignment="Center" VerticalAlignment="Center"/>
                <TextBlock Text="Profit Trend" Grid.Column="5"
HorizontalAlignment="Center" VerticalAlignment="Center"/>
            </Grid>
            <ScrollViewer Grid.Row="1" HorizontalScrollMode="Disabled"
x:Name="scrollViewer">
                <ItemsControl x:Name="RegionSalesListBox" ItemsSource="{Binding
Sales}">
                </ItemsControl>
            </ScrollViewer>
        </Grid>
```

1. Right-click your MainPage.xaml page and select **View Code** from the list. Import the following namespace:

C#

```
using C1.Xaml.Sparkline;
```

Copyright © 2017 GrapeCity, inc. All rights reserved.

1. Add the following code to the **InitializeComponent()** method to create new random data:

C#

```
Random rnd = new Random();
            string[] states = new string[] { "Alabama", "Alaska", "Arizona", "Idaho",
"Illinois", "Indiana", "Ohio", "Oklahoma", "Oregon", "Pennsylvania", "Vermont",
"Virginia", "Washington" };
            for (int i = 0; i < states.Length; i++)
            {
                RegionSalesData rsd = new RegionSalesData();
                rsd.State = states[i];
                rsd.Data = new ObservableCollection<double>();
                for (int j = 0; j < 12; j++)
                {
                    double d = rnd.Next(-50, 50);
                    rsd.Data.Add(d);
                    rsd.NetSales += d;
                    rsd.TotalSales += Math.Abs(d);
                }
                RegionSale sale = new RegionSale(rsd);
                RegionSalesListBox.Items.Add(sale);
            }
```

1. Since you have your MainPage.xaml page set up, right-click your application name and select Add | New Item.
    1. In the **Add New Item** dialog, select **Blank Page** in the right-hand pane.
    2. Name the file **RegionSale** and click **OK**.
2. Right-click your application name again and select **Add | New Item** again.

    1. In the **Add New Item** dialog, select **Code** in the left-hand pane.
    2. Select **Code File** in the right-hand pane.
    3. Name the file **RegionSalesData** and click **OK**.

In this step, you created a new Windows Store application, added the appropriate reference assemblies, and added both another .xaml page and a code file to your application. In the next step, you'll create the **RegionSale.xaml** page that you added in this step.

Step 2 of 4: Creating the RegionSale.xaml Page

In this step, you'll add the markup and code to create the RegionSale.xaml page that you added in Step 1.

1. From the Solution Explorer, double-click RegionSale.xaml to open it.
2. Add the following namespaces to the <Page> tag:

    - xmlns:local="using:RegionSales"
    - xmlns:sp="using:C1.Xaml.Sparkline"

3. Locate the opening <Grid> tag and edit it to resemble the following:

    Markup

    ```
    <Grid Background="#EFEFEF">
    ```

4. Place your cursor between the <Grid> </Grid> tags and insert the following markup. This will add the

Copyright © 2017 GrapeCity, inc. All rights reserved.

necessary column definitions to your application:

Markup

```
<Grid.ColumnDefinitions>
            <ColumnDefinition Width="100"/>
            <ColumnDefinition Width="100"/>
            <ColumnDefinition Width="100"/>
            <ColumnDefinition Width="200"/>
            <ColumnDefinition Width="150"/>
            <ColumnDefinition Width="150"/>
        </Grid.ColumnDefinitions>
```

5. Directly below the column definitions, add the following markup to add the TextBlock controls, C1Sparkline controls, and a Border to correspond with the <Grid> columns:

Markup

```
<TextBlock Text="{Binding State}" Foreground="#444444" FontSize="14"
VerticalAlignment="Center" HorizontalAlignment="Left" Margin="5"
FontFamily="Global User Interface" x:Name="text"/>
        <TextBlock Text="{Binding TotalSalesFormatted}" Grid.Column="1"
FontSize="14" Foreground="#444444" VerticalAlignment="Center"
HorizontalAlignment="Right" Margin="5" FontFamily="Global User Interface"/>
        <TextBlock Text="{Binding NetSales}" Grid.Column="2" FontSize="14"
Foreground="#444444" VerticalAlignment="Center" HorizontalAlignment="Right"
Margin="5" FontFamily="Global User Interface"/>
        <sp:C1Sparkline Grid.Column="3" Height="50" FontFamily="Global User
Interface" Margin="10" x:Name="sparkline"/>
        <sp:C1Sparkline SparklineType="Winloss"  Grid.Column="4" Height="40"
Margin="10" FontFamily="Global User Interface" x:Name="sparklineWinloss" />
        <sp:C1Sparkline SparklineType="Column" Grid.Column="5" Height="50"
FontFamily="Global User Interface" Margin="10" x:Name="sparklineColumn"/>
        <Border Grid.ColumnSpan="6" VerticalAlignment="Bottom"
HorizontalAlignment="Stretch" BorderThickness="1" BorderBrush="#CCCCCC" />
```

6. Right-click the page and select View Code from the list to open the RegionSale.xaml.cs file.
7. Add the following code directly below the **InitializeComponent()** method:

C#

```
this.DataContext = data;
sparkline.Data = data.Data;
sparklineColumn.Data = data.Data;
sparklineWinloss.Data = data.Data;
```

In this step, you added markup and code for the RegionSale.xaml file. In the next step, you'll add code for the RegionSalesData code file.

Step 3 of 4: Creating the RegionSalesData Code File

In this step, you'll add code to the RegionSalesData code file to create the Data Observable Collection.

Copyright © 2017 GrapeCity, inc. All rights reserved.

1. Double-click the RegionSalesData.cs code file to open it.
2. Import the following namespaces:

```
C#
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

2. Below the namespaces, set up your code file's format:

```
C#
namespace RegionSales
{
    public class RegionSalesData
    {
    }

    }
}
```

4. Edit the RegionSalesData class so that it resembles the following sample:

```
C#
public class RegionSalesData
    {
        public ObservableCollection<double> Data { get; set; }
        public string State { get; set; }
        public double TotalSales { get; set; }
        public string TotalSalesFormatted
        {
            get
            {
                return String.Format("{0:c2}", this.TotalSales);
            }
        }
        public double NetSales { get; set; }
    }
```

In this step, you added code to the RegionSalesData code file. In the next step, you'll run your application.

Step 4 of 4: Running the Application

In this step, you'll run your application.

Press **F5** or start debugging to run your application. It should resemble the following image:

Copyright © 2017 GrapeCity, inc. All rights reserved.

Region	Total Sales	Net Sales	Sales Trend	Win/Loss	Profit Trend
Alabama	$342.00	120			
Alaska	$294.00	-78			
Arizona	$367.00	-181			
Idaho	$337.00	151			
Illinois	$290.00	-132			
Indiana	$286.00	-114			
Ohio	$264.00	-28			
Oklahoma	$316.00	88			
Oregon	$359.00	-51			
Pennsylvania	$283.00	29			

Copyright © 2017 GrapeCity, inc. All rights reserved.

Sparkline for UWP Task-Based Help

The task-based help assumes that you are familiar with programming in Visual Studio and know how to use the C1Sparkline control in general. If you are unfamiliar with the **Sparkline for UWP** product, please see the **Sparkline for UWP** Quick Start first.

Each topic in this section provides a solution for specific tasks using the **Sparkline for UWP** product.

Each task-based help topic also assumes that you have created a new Universal Windows application.

Displaying the X-Axis

You can easily display the X-axis for your C1Sparkline control.

In XAML

To display the X-Axis using XAML markup, add the following to your <c1:C1Sparkline> tag:

Markup

```
DisplayXAxis = "True"
```

In Code

To display the X-Axis using code, add the following to your **InitializeComponent()** method:

C#

```
sparkline.DisplayXAxis = true;
```

Using the Date Axis

The X-Axis can be formatted so that **C1Sparkline** Data is displayed over a period of time.

In Code

Use the following code to set the **C1Sparkline**'s DateAxisData property.

Code

```
C1Sparkline.DateAxisData =  new List<DateTime>(){ new DateTime(2013,11,1), new
DateTime(2013,11,2)}
```

Changing C1Sparkline's Appearance

The C1Sparkline control has specific appearance properties which you can use to customize your application. For a full explanation of the appearance properties, please see the C1Sparkline Appearance Properties topic.

The C1Sparkline's Appearance properties can be set in XAML markup or in code.

In Markup

Insert the following markup into the <c1:C1Sparkline> tag to set some of the control's appearance properties:

Markup

```
SeriesColor="#FF4BC128" ShowFirst="True" ShowHigh="True" AxisColor="#FF2859C1".
```

Copyright © 2017 GrapeCity, inc. All rights reserved.

In Code

To set the C1Sparkline's appearance properties, your code should resemble the following:

C#

```
sparkline.MarkersColor = Colors.DeepPink;
sparkline.DisplayXAxis = true;
```

Copyright © 2017 GrapeCity, inc. All rights reserved.