ComponentOne

Sparkline for WPF

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

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

Table of Contents

Sparkline for WPF Overview

ComponentOne Studio introduces **Sparkline for WPF**, a lightweight data visualization control for depicting trends and variation. Designed as an inline chart, Sparkline plots data in a highly condensed way. Unlike standard charts, Sparkline is drawn without basic chart elements like coordinates, axes, legend, and title.

What makes Sparkline chart more interesting to use is their size. Due to compact size, Sparkline charts can be easily embedded in grids and dashboards.

Sparkline Key Features

As a lightweight data visualization control, Sparkline provides various features that makes it suitable for depicting trend lines and variation curves. Here is a quick look at the key features that Sparkline has to offer.

Key Feature	Description
Sparkline Types	The Sparkline control supports three distinct types to cater diverse business needs, including Column, Line, and Winloss.
Markers	To highlight data values, Sparkline supports markers that can be tailored to suit user requirements.
Data Binding	The Sparkline control can easily bind to any enumerable collection of data values.
Axes	Like any other chart, Sparkline supports an x-axis, which can either be displayed or kept hidden as per the user requirement. By default, the x-axis remains hidden in Sparkline. The x-axis can also be used to display data over a span of dates. For more information, refer to **Date Axis**.
Appearance	The **C1Sparkline ('C1Sparkline Class' in the on-line documentation)** class provides a number of properties to customize the control's appearance. In addition to setting markers, users can customize the color of axis, data points (first and last), and the entire series as well.

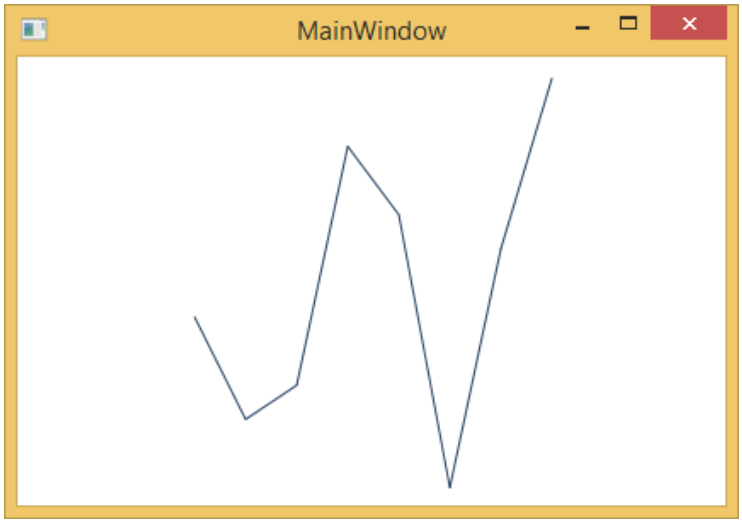
Quick Start: Sparkline for WPF

This quick start section is intended to familiarize you with the Sparkline control. You begin by creating a WPF application in Visual Studio, adding Sparkline control, and populating it with sample data.

Complete the steps given below to see how Sparkline control appears at runtime.

- **Step 1: Setting up the application**
- **Step 2: Adding sample data to populate the control at runtime**
- **Step 3: Adding code to bind the control with sample data**

Here is how a basic Sparkline control appears without any customization.



Step 1: Setting up the application

1. Create a WPF application in Visual Studio.
2. Drag and drop the C1Sparkline control to the MainWindow
3. Edit the XAML view to include relevant namespaces and set some basic properties of the control.

XAML copyCode

```
    <Window x:Class="Sparkline_QuickStart.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
        Title="MainWindow" Height="350" Width="525">
    <Grid>
        <c1:C1Sparkline x:Name="sparkline" Height="250" Width="250"/>
    </Grid>
</Window>
```

Step 2: Adding sample data to populate the control at runtime

In this step, you add a class that returns an enumerable collection of numeric data points to be plotted on the Sparkline chart. This code example assumes that you add a class named **SampleData.cs** to return a collection of numeric values

1. In the Solution Explorer, right-click your project name and select **Add | Class**.
2. Specify the name of the class i.e. SampleData and click **Add**.
3. Add relevant code to create an enumerable collection of numeric data and return the same in a method.
   - **Visual Basic**

```
Public Class SampleData
    Public ReadOnly Property DefaultData() As List(Of Double)
        Get
            Dim data As New List(Of Double)() From { _
                1.0, _
                -2.0, _
                -1.0, _
                6.0, _
                4.0, _
                -4.0, _
                3.0, _
                8.0 _
            }
            Return data
        End Get
    End Property
End Class
```

   - **C#**

```
class SampleData
{

    public List<double> DefaultData
```

```
        {
            get
            {
                List<double> data = new List<double>() { 1.0, -2.0, -1.0, 6.0, 4.0, -4.0, 3.0, 8.0 };
                return data;
            }
        }

    }
```

Step 3: Adding code to bind the control with sample data

The **SampleData.cs** class added in the above step returns a collection of numeric values. In this step, you bind this collection to the Sparkline control so that data can be plotted at runtime. For this, you need to create an object of the **SampleData** class and assign it to the **Data ('Data Property' in the on-line documentation)** property of Sparkline control.

1. Switch to the MainPage.xaml.cs file and add the following import statements.
   - **Visual Basic**

```
Imports C1.WPF.Sparkline
```

   - **C#**

```
using C1.WPF.Sparkline;
```

2. Edit the interaction logic as given below to create an object of the SampleData class and assign it to the control's **Data ('Data Property' in the on-line documentation)** property
   - **Visual Basic**

```
Partial Public Class MainWindow
    Inherits Window
    Private sampleData As New SampleData()
    Public Sub New()
        InitializeComponent()
        sparkline.Data = sampleData.DefaultData
    End Sub
End Class
```

   - **C#**

```
public partial class MainWindow : Window
{
    private SampleData sampleData = new SampleData();
    public MainWindow()
    {
        InitializeComponent();
        sparkline.Data = sampleData.DefaultData;
    }
}
```

3. Press **F5** to run the application and observe how the Sparkline control appears at runtime.

What You Accomplish!

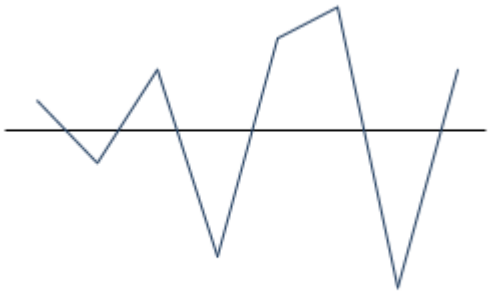
Congratulations! you successfully created a WPF application, added Sparkline control, and bound it to sample data to display a variation in the data points. Sparkline renders as a chart showing variation in data, without the basic chart-like elements like axis, legend, and title.

Features

Axes

By default, the Sparkline control renders without any axis. However, Sparkline supports an x-axis, which can either be displayed or kept hidden as per the user requirements. By default, the x-axis remains hidden in Sparkline.

To display x-axis, you can set the **DisplayXAxis ('DisplayXAxis Property' in the on-line documentation)** property to true. Here is how a Sparkline control with x-axis appears.



Sparkline with x-axis

In XAML

XAML	copyCode

```
<c1:C1Sparkline x:Name="sparkline" Height="250" Width="250" DisplayXAxis="True" />
```

In Code

- **Visual Basic**

```
'Displaying x-axis
sparkline.DisplayXAxis = True
```

- **C#**

```
//Displaying x-axis
sparkline.DisplayXAxis = true;
```

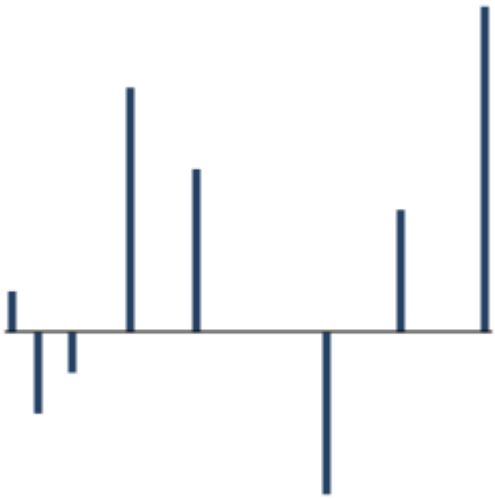
Date Axis

Sparkline supports a date axis to plot data over a span of dates on x-axis. The date axis is nothing but a type of x-axis that allows users to display data at irregular intervals. The date axis takes a collection of dates that acts as x-coordinates and plots data accordingly. This way users can plot data over a span of dates.

For example, consider a scenario where the user wants to display the following monthly sales profit figures as a trend. In the following tabulation, negative values indicate loss.

Dates	Sales Profit (in Hundred Thousands)
01-01-2016	1.0
15-01-2016	-2.0
01-02-2016	-1.0
03-03-2016	6.0
06-04-2016	4.0
11-06-2016	-4.0
19-07-2016	3.0
01-09-2016	8.0

The **C1Sparkline ('C1Sparkline Class' in the on-line documentation)** class provides **DisplayDateAxis ('DisplayDateAxis Property' in the on-line documentation)** property, which if set to true, displays data over a span of dates. Here is how the Sparkline control plots data at irregular intervals.



Complete the steps given below to set the date axis an plot data at irregular intervals.

To configure date axis, you can supply an enumerable collection of dates as x-coordinates and assign it to the **DateAxisData ('DateAxisData Property' in the on-line documentation)** property. This example uses the sample that you created in the **Quick Start** section.

1. Add the following code to the SampleData.cs class that you created in the Quick Start.
    - **Visual Basic**

```
Public ReadOnly Property DefaultDateAxisData() As List(Of DateTime)
    Get
        Dim dates As New List(Of DateTime)()
        dates.Add(New DateTime(2016, 1, 1))
        dates.Add(New DateTime(2016, 1, 15))
        dates.Add(New DateTime(2016, 2, 1))
        dates.Add(New DateTime(2016, 3, 3))
        dates.Add(New DateTime(2016, 4, 6))
        dates.Add(New DateTime(2016, 6, 11))
        dates.Add(New DateTime(2016, 7, 19))
        dates.Add(New DateTime(2016, 9, 1))
        Return dates
    End Get
End Property
```

    - **C#**

```
public List<DateTime> DefaultDateAxisData
{
    get
    {
        List<DateTime> dates = new List<DateTime>();
        dates.Add(new DateTime(2016, 1, 1));
        dates.Add(new DateTime(2016, 1, 15));
        dates.Add(new DateTime(2016, 2, 1));
        dates.Add(new DateTime(2016, 3, 3));
        dates.Add(new DateTime(2016, 4, 6));
        dates.Add(new DateTime(2016, 6, 11));
        dates.Add(new DateTime(2016, 7, 19));
        dates.Add(new DateTime(2016, 9, 1));
        return dates;
    }
}
```

2. Switch to the MainWindow.xaml.cs class, and edit the interaction logic to set sparkline type and assign the date collection to the **DateAxisData ('DateAxisData Property' in the on-line documentation)** property.
    - **Visual Basic**

```
'Display Date Axis
sparkline.Data = sampleData.DefaultData
sparkline.SparklineType = SparklineType.Column
sparkline.DateAxisData = sampleData.DefaultDateAxisData
sparkline.DisplayXAxis = True
```

    - **C#**

```
//Display Date Axis
sparkline.Data = sampleData.DefaultData;
sparkline.SparklineType = SparklineType.Column;
sparkline.DateAxisData = sampleData.DefaultDateAxisData;
sparkline.DisplayXAxis = true;
```

Data Binding

Data forms the core for any data visualization control. The Sparkline control can be bound to any enumerable collection of numeric data values. Since Sparkline supports a date axis, it can also be bound to a collection of dates that acts as x-coordinates.

The **C1Sparkline ('C1Sparkline Class' in the on-line documentation)** class provides two properties to bind the sparkline control with data.

Property	Description
Data ('Data Property' in the on-line documentation)	This property binds the sparkline control to a collection of numeric values.
DateAxisData ('DateAxisData Property' in the on-line documentation)	This property binds the sparkline control to a collection of dates.

Refer to **Quick Start** and **Date Axis** topics to see how data binding can be achieved by setting the above properties.

Sparkline Types

Sparkline supports three different chart types, namely Line, Column and Winloss, for visualizing data in different context. For example, Line charts are suitable to visualize continuous data, while Column sparklines are used in scenarios where data comparison is involved. Similarly, a Win-Loss sparkline is best used to visualize a true-false (that is, win-loss) scenario.

Line Sparkline

Line sparkline is drawn as a linear graph that best suits scenarios where a user wants to visualize some trend in the data. Commonly, this type of sparkline is used to visualize sales figures or stock values.

Here is how a Sparkline control with **SparklineType ('SparklineType Property' in the on-line documentation)** property set to Line appears.

By default, Sparkline renders as a Line sparkline.

Column Sparkline

Column sparkline renders data points in columns. This type of sparkline is suitable to visualize comparison in data. In Column Sparkline, positive data points are drawn in upward direction, while negative data points are drawn in downward direction.

Here is how a Sparkline control with **SparklineType ('SparklineType Property' in the on-line documentation)** property set to Column appears.

Win-Loss Sparkline

Win-Loss sparkline displays data points through equal-sized columns drawn in upward and downward direction. This type of Sparkline is used to visualize win/profit-loss scenarios. Columns drawn in upward direction indicate a win, while downward columns indicate a loss.

Here is how a Sparkline control with **SparklineType ('SparklineType Property' in the on-line documentation)** property set to Win-Loss appears.

Setting Sparkline Type

The **C1Sparkline ('C1Sparkline Class' in the on-line documentation)** class provides **SparklineType ('SparklineType Property' in the on-line documentation)** property to set the chart type in XAML or code.
The **SparklineType ('SparklineType Property' in the on-line documentation)** property accepts the following values from the **SparklineType ('SparklineType Enumeration' in the on-line documentation)** enumeration:

- **Column** - Allows you to specify the column sparkline.
- **Line** - Allows you to specify the line sparkline.
- **Winloss** - Allows you to specify the winloss sparkline.

In XAML

To set the chart type for Sparkline in XAML, you need to set the **SparklineType ('SparklineType Property' in the on-line documentation)** property to a specific chart type in XAML.

XAML copyCode

```
<c1:C1Sparkline x:Name="sparkline" Height="150" Width="250" SparklineType="Column" />
```

In Code

You can also set the chart type for Sparkline in code using the SparklineType property.

- **Visual Basic**

```
'Setting the Chart Type
sparkline.SparklineType = SparklineType.Column
```

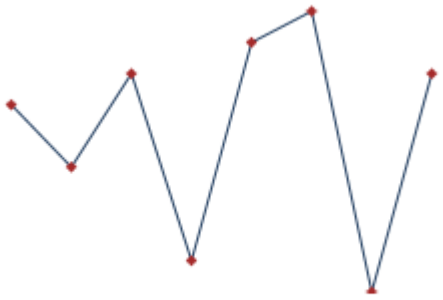
- **C#**

```
//Setting the Chart Type
sparkline.SparklineType = SparklineType.Column;
```

Markers

Sparkline control is primarily designed to depict trends and variation by plotting a set of data points. Depending on the Sparkline type, the data either gets displayed as a linear graph, winloss trend, or as columns without any chart-specific elements or customization. However, users might need to customize the way data points appear. For example, they may need to indicate or highlight the data points on a Sparkline. For this, Sparkline comes with markers whose visibility can be explicitly set to highlight various data points.

The **C1Sparkline ('C1Sparkline Class' in the on-line documentation)** class provides **ShowMarkers ('ShowMarkers Property' in the on-line documentation)** property, which if set to true highlights all the data points. Here is how the Sparkline control looks with markers highlighting all the data points.



Sparkline with Markers highlighting Data Points

- **Visual Basic**

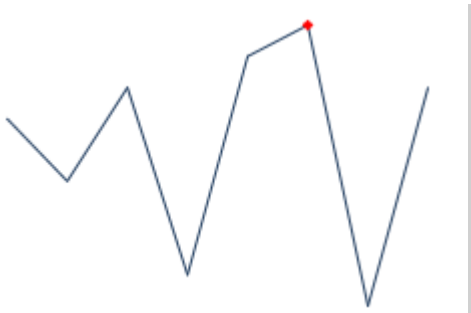
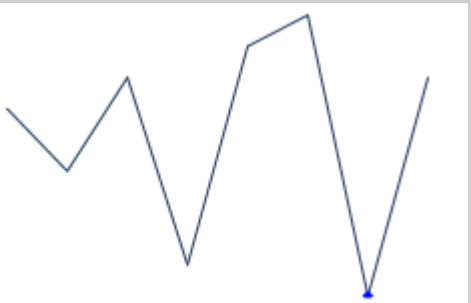
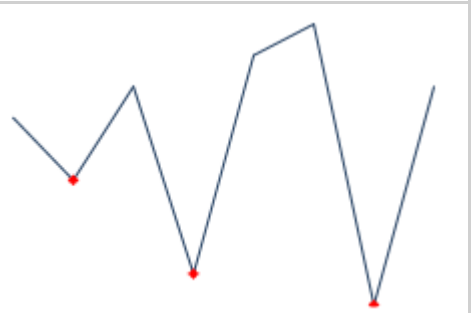
```
'Displaying Markers
sparkline.ShowMarkers = True
```

- **C#**

```
//Displaying Markers
sparkline.ShowMarkers = true;
```

The **ShowMarkers ('ShowMarkers Property' in the on-line documentation)** property highlights all the data points in Sparkline. However, some users might want to highlight some specific values to suit their business needs. For this, the Sparkline control allows users to apply markers on specific data points as well. The following properties can be set in code for the same.

Property	Description	Output
ShowFirst ('ShowFirst Property' in the on-line documentation)	You can set this property to true in code to highlight the first data point in the output as shown in the image alongside. By default, the marker highlights the first data point in maroon color.	
ShowLast ('ShowLast Property' in the on-line documentation)	You can set this property to true in code to highlight the last data point in the output as shown in the image alongside. By default, the marker highlights the last data point in green color.	

ShowHigh ('ShowHigh Property' in the on-line documentation)	You can set this property to true in code to highlight the highest value in the output as shown in the image alongside. By default, the marker highlights the highest data point in red color.	
ShowLow ('ShowLow Property' in the on-line documentation)	You can set this property to true in code to highlight the lowest value in the output as shown in the image alongside. By default, the marker highlights the lowest data point in blue color.	
ShowNegative ('ShowNegative Property' in the on-line documentation)	You can set this property to true in code to highlight all the negative data points in the output as shown in the image alongside. By default, the marker highlights the negative data point(s) in red color.	

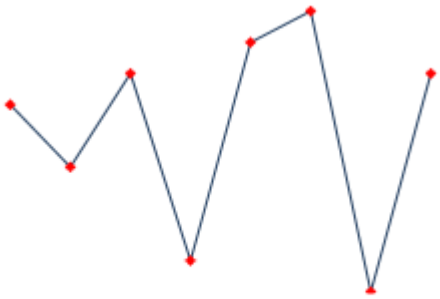
Note that markers can not be used to highlight data points in Column and WinLoss sparklines.

Customizing Marker Color

Markers indicate or highlight various data points in a default color. For example, negative data points get highlighted in red color on setting **ShowNegative ('ShowNegative Property' in the on-line documentation)** property to true, while the last data point gets highlighted in green color when the **ShowLast ('ShowLast Property' in the on-line documentation)** property is set to true. However, you can modify this default behavior and highlight data points in a color of your choice, making Sparkline more customizable for users from appearance perspective.

The **C1Sparkline ('C1Sparkline Class' in the on-line documentation)** class provides various properties to customize marker colors in code. For example, the **MarkersColor ('MarkersColor Property' in the on-line documentation)** property specifies the color of the markers for each data point.

Here is how a Sparkline control appears after setting the **MarkersColor ('MarkersColor Property' in the on-line documentation)** property to Red.



Markers highlighting data points in Red Color

To highlight data markers in a different color programmatically, set the MarkersColor property in code.

- **Visual Basic**

```
'Setting Marker Color
sparkline.MarkersColor = Colors.Red
```

- **C#**

```
//Setting Marker Color
sparkline.MarkersColor = Colors.Red;
```

Here is a list of properties that can be used to customize marker colors for highlighting specific data points.

Property	Description
FirstMarkerColor ('FirstMarkerColor Property' in the on-line documentation)	This property can be set to specify the marker color for the first data point in Sparkline.
LastMarkerColor ('LastMarkerColor Property' in the on-line documentation)	This property can be set to specify the marker color for the last data point in Sparkline.
HighMarkerColor ('HighMarkerColor Property' in the on-line documentation)	This property can be set to specify the marker color for the highest data point in Sparkline.
LowMarkerColor ('LowMarkerColor Property' in the on-line documentation)	This property can be set to specify the marker color for the lowest data point in Sparkline.
NegativeColor ('NegativeColor Property' in the on-line documentation)	This property can be set to specify the marker color for negative data points in Sparkline.

Sparkline Task-Based Help

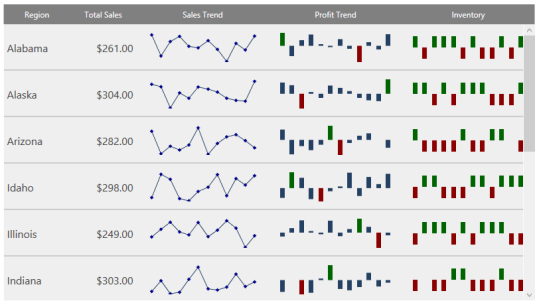
Using Sparkline in Data Management Controls

This section walks you through adding sparklines in data management controls like ItemsControl, ListBox or data grids. You begin with creating a new WPF application, adding code in the XAML view to display a grid-like layout with five column fields, and adding the interaction logic in the code view to display data.

Complete the following steps to add a group of sparkline in grid as shown in the given image.

- **Step 1: Setting up the Application**
- **Step 2: Adding code to display data and Sparkline in XAML grid**

- **Step 3: Running the Application**



Step 1: Setting up the Application

1. Create a WPF application in Visual Studio.
2. Drag and Drop the C1Sparkline control onto the MainWindow.
3. Edit the XAML view to include ComponentOne namespace.

XAML

```
xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
```

4. Replace the grid tags with the given code to create a XAML grid with five columns displaying different data fields namely Region, Total Sales, Sales Trend, Profit Trend and Inventory.

XAML copyCode

```
<Grid>
    <Grid.Resources>
        <Style TargetType="TextBlock">
            <Setter Property="FontSize" Value="12" />
        </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="200"/>
                <ColumnDefinition Width="200"/>
                <ColumnDefinition Width="200"/>
            </Grid.ColumnDefinitions>
            <TextBlock Text="Region" Foreground="White" HorizontalAlignment="Center" VerticalAlignment="Center"/>
            <TextBlock Text="Total Sales" Grid.Column="1" Foreground="White" HorizontalAlignment="Center" VerticalAlignment="Center"/>
            <TextBlock Text="Sales Trend" Grid.Column="2" Foreground="White" HorizontalAlignment="Center" VerticalAlignment="Center"/>
            <TextBlock Text="Profit Trend" Grid.Column="3" Foreground="White" HorizontalAlignment="Center" VerticalAlignment="Center"/>
            <TextBlock Text="Inventory" Grid.Column="4" Foreground="White" HorizontalAlignment="Center" VerticalAlignment="Center"/>

        </Grid>
        <ScrollViewer Grid.Row="1">
            <ItemsControl x:Name="RegionSalesListBox" ItemsSource="{Binding Sales}" >
                <ItemsControl.ItemTemplate>
                    <DataTemplate>
                        <Grid Background="#EFEFEF">
                            <Grid.ColumnDefinitions>
                                <ColumnDefinition Width="100"/>
                                <ColumnDefinition Width="100"/>
                                <ColumnDefinition Width="200"/>
                                <ColumnDefinition Width="200"/>
                                <ColumnDefinition Width="200"/>
                            </Grid.ColumnDefinitions>
                            <TextBlock Text="{Binding State}" Foreground="#444444" FontSize="16" VerticalAlignment="Center" HorizontalAlignment="Left" Margin="5" />
                            <TextBlock Text="{Binding TotalSalesFormatted}" Grid.Column="1" FontSize="16" Foreground="#444444" VerticalAlignment="Center" HorizontalAlignment="Right" Margin="5"/>
                            <c1:C1Sparkline Data="{Binding Data}" Grid.Column="2" Height="50" Margin="10" ShowMarkers="True" MarkersColor="DarkBlue"/>
                            <c1:C1Sparkline Data="{Binding Data}" SparklineType="Column" Grid.Column="3" Height="50" Margin="10" ShowHigh="True" ShowLow="True" LowMarkerColor="DarkRed" HighMarkerColor="DarkGreen"/>
                            <c1:C1Sparkline Data="{Binding Data}" SparklineType="Winloss" Grid.Column="4" Height="40" Margin="10" SeriesColor="DarkGreen" NegativeColor="DarkRed" ShowNegative="True"/>
                            <Border Grid.ColumnSpan="6" VerticalAlignment="Bottom" HorizontalAlignment="Stretch" BorderThickness="1" BorderBrush="#CCCCCC" />
                        </Grid>
                    </DataTemplate>
                </ItemsControl.ItemTemplate>
            </ItemsControl>
        </ScrollViewer>
    </Grid>
</Grid>
```

Step 2: Adding code to display data and Sparkline in XAML grid

1. Switch to the code view, that is MainWindow.xaml.cs file.
2. Add the following import statement.
   - **Visual Basic**

```
Imports System.Collections.ObjectModel
```

   - **C#**

```
using System.Collections.ObjectModel;
```

3. Edit the interaction logic for XAML as given below to display data and sparkline in the grid.
   - **Visual Basic**

```
Partial Public Class MainWindow
    Inherits Window
    Public Property Sales() As ObservableCollection(Of RegionSalesData)
        Get
            Return m_Sales
        End Get
        Private Set(value As ObservableCollection(Of RegionSalesData))
            m_Sales = Value
        End Set
    End Property
    Private m_Sales As ObservableCollection(Of RegionSalesData)

    Public Sub New()
        Dim rnd As New Random()
        Dim states As String() = New String() {"Alabama", "Alaska", "Arizona", "Idaho", "Illinois", "Indiana", _
            "Ohio", "Oklahoma", "Oregon", "Pennsylvania", "Vermont", "Virginia", _
            "Washington"}
        Me.Sales = New ObservableCollection(Of RegionSalesData)()
        For i As Integer = 0 To states.Length - 1
            Dim rsd As New RegionSalesData()
            rsd.State = states(i)
            rsd.Data = New ObservableCollection(Of Double)()
            For j As Integer = 0 To 11
                Dim d As Double = rnd.[Next](-50, 50)
                rsd.Data.Add(d)
                rsd.TotalSales += Math.Abs(d)
            Next
            Me.Sales.Add(rsd)
        Next

        Me.DataContext = Me
    End Sub
End Class

Public Class RegionSalesData
    Public Property Data() As ObservableCollection(Of Double)
        Get
            Return m_Data
        End Get
        Set(value As ObservableCollection(Of Double))
            m_Data = Value
        End Set
    End Property
```

```
    Private m_Data As ObservableCollection(Of Double)
    Public Property State() As String
        Get
            Return m_State
        End Get
        Set(value As String)
            m_State = Value
        End Set
    End Property
    Private m_State As String
    Public Property TotalSales() As Double
        Get
            Return m_TotalSales
        End Get
        Set(value As Double)
            m_TotalSales = Value
        End Set
    End Property
    Private m_TotalSales As Double
    Public ReadOnly Property TotalSalesFormatted() As String
        Get
            Return [String].Format("{0:c2}", Me.TotalSales)
        End Get
    End Property
End Class
```

- **C#**

```
public partial class MainWindow : Window
{
    public ObservableCollection<RegionSalesData> Sales { get; private set; }

    public MainWindow()
    {
        Random rnd = new Random();
        string[] states = new string[] { "Alabama", "Alaska", "Arizona", "Idaho", "Illinois", "Indiana", "Ohio", "Oklahoma", "Oregon", "Pennsylvania", "Vermont", "Virginia", "Washington" };
        this.Sales = new ObservableCollection<RegionSalesData>();
        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.TotalSales += Math.Abs(d);
            }
            this.Sales.Add(rsd);
        }

        this.DataContext = this;
    }
}

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);
        }
    }
}
```

Step 3: Running the Application

1. Debug your application to see if any error exist.
2. Press **F5** to run your application and see the output.