Sparkline for UWP

2018.03.07 更新

グレープシティ株式会社

Sparkline for UWP

目次

Sparkline for UWP

Sparkline for UWP は、データを簡単に視覚化できる軽量なチャートコントロールです。スパークラインは、データテンプレートやダッシュボードタイルなどの小さなスペースでトレンドを見せる方法として優れています。

主な特長

Sparkline for UWP には、次の機能があります。

- **3つのグラフタイプをサポート**
  C1Sparkline コントロールは、**Column**、**Line**、**WinLoss** の3つのグラフタイプをサポートしています。
- **マーカーポイントを色付きで表示**
  スパークラインは、マーカーポイントを色付きで表示することができます。最大、最小、負、最初、最後のポイントの色をそれぞれ設定できます。
- **柔軟性の高いデータ連結**
  数値データの列挙可能コレクションに **C1Sparkline** を連結します。スパークラインにコードでデータを入力したり、**XAML** でデータ連結を設定することができます。
- **日付軸のサポート**
  C1Sparkline コントロールでは、水平軸と垂直軸がサポートされるだけでなく、日付軸を設定することができます。それには、x 座標となる一連の日付を指定します。

クイックスタート

このクイックスタートガイドは、**Sparkline for UWP** を初めて使用するユーザーのために用意されています。このクイックスタートでは、Visual Studio で新しいプロジェクトを作成し、アプリケーションに **Sparkline for UWP** コントロールを追加して、コントロールの外観と動作をカスタマイズします。

手順1：アプリケーションの作成

この手順では、Visual Studio を使用して新しい UWP スタイルのアプリケーションを作成し、ページに C1Sparkline コントロールを追加します。

1. Visual Studio で、[**ファイル**]→[**新規作成**]→[**プロジェクト**]を選択します。
2. [**新しいプロジェクト**]ダイアログボックスで、次の操作を実行します。
   a. 左側のペインで言語を展開します。
   b. 言語の下で、[**Windows ストア**]を選択します。
   c. テンプレートリストで、[**新しいアプリケーション (XAML)**]を選択します。
   d. 名前を入力し、[**OK**]をクリックしてプロジェクトを作成します。
3. ソリューションエクスプローラで[**参照**]フォルダを右クリックし、ドロップダウンリストから[**参照の追加**]を選択します。[**参照マネージャー**]が開きます。
4. [**参照マネージャー**]の左側のペインで、[**Windows**]の横にあるドロップダウン矢印を選択します。[**拡張**]を選択します。
5. [**C1.Xaml**]参照と[**C1.Xaml.Sparkline**]参照のチェックボックスをオンにし、[**OK**]をクリックします。
6. 次のマークアップを開始タグ `<Page>` に追加します。

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

   `<Page>` タグは次のようになります。

```
<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. `<Grid> </Grid>` タグの間にカーソルを置きます。
8. Visual Studio ツールボックスで **C1Sparkline** コントロールを見つけます。コントロールをダブルクリックしてアプリケーションに追加します。
9. 次のサンプルのように、タグ `<c1:C1Sparkline/>` を編集します。これで、コントロールをコードで呼び出すための名前が追加され、スパークラインコントロールの色がカスタマイズされます。また、適切な連結ステートメントも追加されます。

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

ここまでの成果

この手順では、新しい Windows ストアアプリケーションを作成し、適切なアセンブリ参照をアプリケーションに追加して、C1Sparkline コントロールを追加しました。

手順2：コードの追加

この手順では、C1Sparkline コントロールに連結するデータを指定するためのコードを追加します。

1. **MainPage.xaml** を右クリックし、リストから[**コードの表示**]を選択します。
2. 次のコードを **InitializeComponent()** メソッドの下に追加して、マーカーの色を変更し、C1Sparkline コントロールのデータを作成します。

```
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. 次に、手順2で追加したコードの下に、次のクラスを追加します。

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

4. 次のコードを追加して、スパークラインにデータを挿入します。

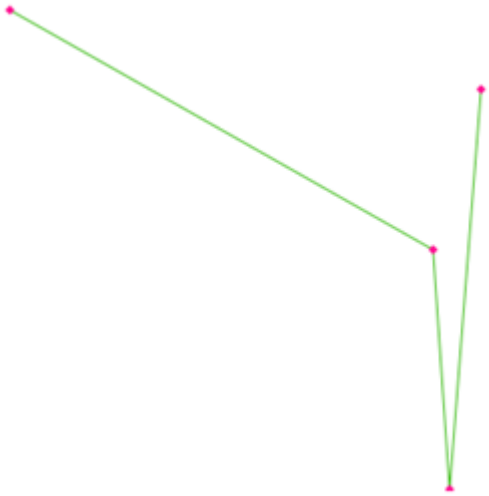
```
public class Sales
    {
        public List<double> Data{get;set;}
        public List<DateTime> DateAxis{get;set;}

        public Sales()
        {
            Data = new List<double>();
            DateAxis = new List<DateTime>();
        }
    }
```

手順3：アプリケーションの実行

この手順では、作成したアプリケーションを実行します。

[F5]キーを押すか、デバッグを開始して、アプリケーションを実行します。次の図のようになります。



SeriesColor（色）は明るい緑色、マーカーはピンク色で示されていることに注意してください。

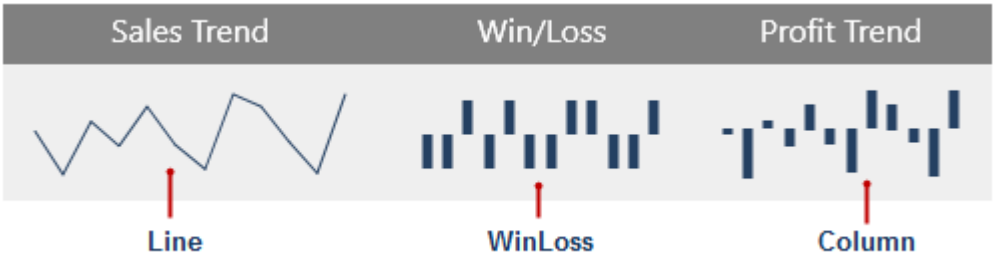
ここまでの成果

おめでとうございます。これで、**Sparkline for UWP** クイックスタートは終了です。スパークラインアプリケーションを作成し、データを C1Sparkline コントロールに追加して、グラフの外観をカスタマイズしました。

C1Sparkline for UWPの概念と主なプロパティ

以下のトピックでは、**Sparkline for UWP** コントロールのプロパティの基本概念について説明します。これらのプロパティを使用して、軸やコントロールの外観からグラフに含まれるデータに至るまで C1Sparkline コントロールを詳細にカスタマイズできます。

C1Sparkline コントロールを使用すると、データに並べて、簡潔で公正なデータ分析を表示することができます。3つのタイプの **C1Sparkline** があるため、さまざまなタイプのデータを柔軟にグラフィカル表示することができます。次の図は、この3つのタイプの **C1Sparkline** コントロールを示しています。



標準的なスパークラインの作成手順は次のとおりです。

1. スパークラインタイプ（**SparklineType** プロパティ）を選択します。
   C1Sparkline は3つのタイプ（Column、Line、WinLoss）をサポートしています。最適なタイプはデータの性質に大きく依存します。これについては、後で説明します。
2. 軸を設定します。
   通常、軸を設定するには、使用する X 軸の表示/非表示とタイプ、垂直軸のタイプ、最小値、および最大値を指定します。
3. データ系列の追加します（**Data**）。
   ここでは、データを作成して連結するか、外部データソースに連結します。
4. **Appearance** プロパティを使用して、グラフの外観を調整します。

軸

次のプロパティを使用して、C1Sparkline の軸を表現します。

プロパティ	説明	コードサンプル
DisplayDateAxis	このプロパティを true に設定すると、データが一定期間にわたって X 軸に表示されます。このタイプの軸を使用すると、特定の日付にデータポイントがない場合に、データにギャップを挿入することができます。	`C1Sparkline.DisplayDateAxis = true`
DisplayXAxis	このプロパティを true または false に設定して、X 軸を表示するかどうかを示すことができます。	`C1Sparkline.DisplayXAxis = true`
ManualMax	スパークライングループ内のすべてのスパークラインで共有する垂直軸の最大値を取得または設定します。maxAxisType が custom に設定されていない場合は、この値を 0 に設定する必要があります。	`C1Sparkline.ManualMax = 20`
ManualMin	スパークライングループ内のすべてのスパークラインで共有する垂直軸の最小値を取得または設定します。minAxisType が custom に設定されていない場合は、この値を 0 に設定する必要があります。	`C1Sparkline.ManualMin = -5`
MaxAxisType	このプロパティは、スパークライングループ内のスパークライン	`C1Sparkline.MaxAxisType =`

	の垂直軸の最大をどのように計算するかを指定します。	`SparklineAxisMinMax.Group`
MinAxisType	このプロパティは、スパークライングループ内のスパークラインの垂直軸の最小をどのように計算するかを指定します。	`C1Sparkline.MinAxisType = SparklineAxisMinMax.Group`

データとデータの連結

このセクションでは、データとデータ連結について説明します。

Data と **DateAxisData** の2つのプロパティで、C1Sparkline コントロールで使用されるデータを制御します。次の表に、これらのプロパティの詳細を示します。

プロパティ	説明	コードサンプル
Data	C1Sparkline の値を提供します。	`C1Sparkline.Data = new List<double>(){1,2,3,4,-1,-3,-4.5,6}`
DateAxisData	C1Sparkline の日時軸の値を提供します。	`C1Sparkline.DateAxisData = new List<DateTime>(){ new DateTime(2013,11,1), new DateTime(2013,11,2)}`

データ連結グラフの作成に必要な手順は、次のとおりです。

1. スパークラインタイプ（SparklineType プロパティ）を選択します。
2. Binding プロパティを目的のスパークラインデータを含む項目のコレクションに設定します。この設定については、「**クイックスタート**」で説明しています。
3. Appearance プロパティを使用して、グラフの外観を調整します。

Sparkline のタイプ

C1Sparkline コントロールを使用して、データをグラフィカルに表現する小さなインライングラフを作成することができます。

Column スパークラインを設定するには、**SparklineType** プロパティに文字列 Column を指定します。

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

使用できる C1Sparkline のタイプは、**SparklineType** 列挙のメンバで指定します。

Column スパークライン

Column C1Sparkline は、それまでの値と現在の値が直接には関連していないデータを表す場合に便利です。Column C1Sparkline は、スポーツのスコアやレジのレシートなど、連続性のないデータを表す場合に最適です。次の図は、Type プロパティを Column に設定した場合の C1Sparkline コントロールを示しています。

Line スパークライン

Line C1Sparkline は、株価や売上データなど、連続的なデータを表す場合に便利です。次の図は、**SparklineType** プロパティを Line に設定した場合の C1Sparkline コントロールを示しています。

WinLoss スパークライン

WinLoss タイプの C1Sparkline コントロールは、true/false や勝敗データを表す場合に便利です。Column タイプの C1Sparkline とは異なり、WinLoss の棒はすべて同じサイズです。

スパークラインの外観

組み込みの色プロパティを使用すると、C1Sparkline コントロールを簡単にカスタマイズできます。

テキストのプロパティ

以下のプロパティを使用すると、C1Sparkline コントロールのテキストの外観をカスタマイズできます。

プロパティ	説明
FontFamily	コントロールのフォントファミリを取得または設定します。これは依存プロパティです。
FontSize	フォントサイズを取得または設定します。これは依存プロパティです。
FontStretch	フォントを画面上で伸縮する比率を取得または設定します。これは依存プロパティです。
FontStyle	フォントスタイルを取得または設定します。これは依存プロパティです。
FontWeight	指定されたフォントの太さを取得または設定します。これは依存プロパティです。

色のプロパティ

次のプロパティを使用して、コントロール自体に使用される色をカスタマイズできます。

プロパティ	説明
Background	コントロールの背景を描画するブラシを取得または設定します。これは依存プロパティです。
Foreground	前景色を描画するブラシを取得または設定します。これは依存プロパティです。

境界線のプロパティ

次のプロパティを使用して、コントロールの境界線をカスタマイズできます。

プロパティ	説明
BorderBrush	コントロールの境界線の背景を描画するブラシを取得または設定します。これは依存プロパティです。

BorderThickness	コントロールの境界線の太さを取得または設定します。これは依存プロパティです。

C1Sparkline の外観のプロパティ

C1Sparkline コントロールでは、さまざまなプロパティを使用して外観を変更することができます。作成するスパークラインのタイプ、マーカーと軸の表示/非表示、軸、マーカー、系列に適用する色、およびデータの方向を決定することができます。

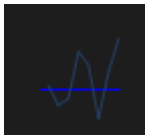
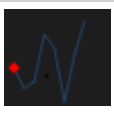
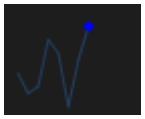
次の表に、一般的な2つの外観プロパティの詳細を示します。

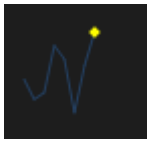
プロパティ名	プロパティの説明	コードサンプル
RightToLeft	データを右から左に表示することができます。デフォルトでは、データは左から右に表示されます。	`C1Sparkline.RightToLeft = true`
SparklineType	スパークラインコントロールのタイプを決定します。	`C1Sparkline.SparklineType = SparklineType.Column`

次の表に、表示/非表示プロパティの詳細を示します。

プロパティ名	プロパティの説明	コードサンプル
ShowFirst	最初のデータマーカーの表示/非表示を決定します。	`C1Sparkline.ShowFirst = true`
ShowLast	最後のデータマーカーの表示/非表示を決定します。	`C1Sparkline.ShowLast = true`
ShowLow	最小データマーカーの表示/非表示を決定します。	`C1.Sparkline.ShowLow = true`
ShowHigh	最大データマーカーの表示/非表示を決定します。	`C1Sparkline.ShowHigh = true`
ShowMarkers	すべてのデータマーカーの表示/非表示を決定します。	`C1Sparkline.ShowMarkers = true`
ShowNegative	負のデータマーカーの表示/非表示を決定します。	`C1Sparkline.ShowNegative = true`
DisplayEmptyCellsAs	空のセルの表示方法を決定します。	`C1Sparkline.DisplayEmptyCellsAs= EmptyValueStyle.Gaps`

次の表に、色プロパティの詳細を示します。

プロパティ名	プロパティの説明	コードサンプル	画像
AxisColor	X 軸の色を取得または設定します。 **DisplayXAxis** プロパティは **true** に設定する必要があります。	`AxisColor = Colors.Red`	
FirstMarkerColor	スパークライングループ内の各スパークラインの最初のデータポイントの色を取得または設定します。 **ShowFirst** プロパティは **true** に設定する必要があります。	`FirstMarkerColor = Colors.Red`	
HighMarkerColor	スパークライングループ内の各スパークラインの最大データポイントの色を取得または設定します。 **ShowHigh** プロパティは **true** に設定する必要があります。	`HighMarkerColor = Colors.Blue`	

LastMarkerColor	スパークライングループ内の各スパークラインの最後のデータポイントの色を取得または設定します。 **ShowLast** プロパティは **true** に設定する必要があります。	`LastMarkerColor = Colors.Yellow`	
LowMarkerColor	スパークライングループ内の各スパークラインの最小データポイントの色を取得または設定します。 **ShowLow** プロパティは **true** に設定する必要があります。	`LowMarkerColor = Colors.Red`	
MarkersColor	スパークライングループ内の各スパークラインのデータマーカーの色を指定します。 **ShowMarkers** プロパティは **true** に設定する必要があります。	`MarkersColor = Colors.DeepPink`	
NegativeColor	スパークライングループ内の各スパークラインの負のデータポイントの色を指定します。 **ShowNegative** プロパティは **true** に設定する必要があります。	`NegativeColor = Colors.Pink`	
SeriesColor	折れ線グラフの線の色を指定します。このプロパティは、**Line** グラフタイプの **C1Sparkline** コントロールでのみ設定できます。	`SeriesColor = Colors.Brown`	

チュートリアル

このチュートリアルは、読者が Visual Studio のプログラミングに精通していることを前提としています。ただし、**Sparkline for UWP** については、手順を追って説明されており、予備知識は特に必要ありません。このセクションに示される手順に従って作業を進めるだけで、**Sparkline for UWP** の機能を具体的に示すプロジェクトを作成できます。

チュートリアルプロジェクトを実行し、自分で変更してみてください。

ListBox への C1Sparkline の追加

このチュートリアルでは、ListBox コントロールにデータをグラフで示す C1Sparkline コントロールを作成します。

手順1：アプリケーションの作成

この手順では、Visual Studio で新しい Windows ストアアプリケーションを作成し、アセンブリ参照を追加して、アプリケーションに必要な .xaml ファイルとコードファイルを追加します。

1. [ファイル]→[新規作成]→[プロジェクト]を選択し、[**新しいプロジェクト**]ダイアログボックスを開きます。
   a. 右側のペインで C# の下にある[Windows ストア]を選択します。
   b. 左側のペインで**[新しいアプリケーション (XAML)]**を選択します。
   c. アプリケーションの名前を入力し（この場合は、**RegionSales**）、[**OK**]をクリックします。新しい空の Windows ストアアプリケーションが開きます。
2. ソリューションエクスプローラで、[**参照**]ファイルを右クリックし、リストから[**参照の追加**]を選択します。次のアセンブリ参照を参照して選択します。
   - C1.Xaml.dll
   - C1.Xaml.Sparkline.dll
3. **MainPage.xaml** ファイルをダブルクリックして開きます。
4. ページ先頭の `<Page>` タグに次の名前空間宣言を追加します。
   - C1.Xaml.dll
   - C1.Xaml.Sparkline.dll

   ページ先頭の`<Page>` タグは次のサンプルのようになります。

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">
```

5. `<Grid> </Grid>` タグの間にカーソルを置き、次のマークアップを挿入します。これで、グリッドのリソース、行、列の定義がそれぞれ作成されます。

```
<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>
```

6. </Grid.ColumnDefinitions> タグのすぐ後に次のマークアップを追加して、いくつかのラベル TextBox コントロールと、RegionSalesListBox を含む別の ScrollViewer コントロールを作成します。

```
<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>
```

7. MainPage.xaml ページを右クリックし、リストから[**コードの表示**]を選択します。次の名前空間をインポートします。

▶ C#コードの書き方

C#

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

8. 次のコードを **InitializeComponent()** メソッドに追加して、新しいランダムデータを作成します。

▶ C#コードの書き方

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);
            }
```

9. MainPage.xaml ページの設定が終了したので、アプリケーション名を右クリックし、[**追加**]→[**新しい項目**]を選択します。
   a. [**新しい項目の追加**]ダイアログで、右側のペインの[**空白のページ**]を選択します。
   b. ファイルに **RegionSale** という名前を付け、[**OK**]をクリックします。
10. アプリケーション名をもう一度右クリックし、[**追加**]→[**新しい項目**]を選択します。
    a. [**新しい項目の追加**]ダイアログで、左側のペインの[コード]を選択します。
    b. 右側のペインの[**コードファイル**]を選択します。
    c. ファイルに **RegionSalesData** という名前を付け、[**OK**]をクリックします。

この手順では、新しい Windows ストアアプリケーションを作成し、適切な参照アセンブリを追加し、もう1つの .xaml ページとコードファイルをアプリケーションに追加しました。次の手順では、この手順で追加した **RegionSale.xaml** ページを作成します。

手順2：RegionSale.xaml ページの作成

この手順では、手順1で追加した **RegionSale.xaml** ページを作成するためのマークアップとコードを追加します。

1. ソリューションエクスプローラで、**RegionSale.xaml** をダブルクリックして開きます。
2. 次の名前空間をタグ `<Page>` に追加します。
   - xmlns:local="using:RegionSales"
   - xmlns:sp="using:C1.Xaml.Sparkline"
3. 開始タグ `<Grid>` を見つけて、次のように編集します。

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

4. `<Grid> </Grid>`タグの間にカーソルを置き、次のマークアップを挿入します。これで、必要な列定義がアプリケーションに追加されました。

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

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

5. 列定義のすぐ下に次のマークアップを追加して、<Grid> 列に対応する TextBlock コントロール、C1Sparkline コントロール、および Border を追加します。

```
<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. ページを右クリックし、リストから[**コードの表示**]を選択して **RegionSale.xaml.cs** ファイルを開きます。
7. **InitializeComponent()** メソッドのすぐ下に、次のコードを追加します。

▸ C# コードの書き方

C#

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

この手順では、RegionSale.xaml ファイルのマークアップとコードを追加しました。次の手順では、RegionSalesData コードファイルのコードを追加します。

手順3：RegionSalesData コードファイルの作成

この手順では、データの ObservableCollection を作成するためのコードを RegionSalesData コードファイルに追加します。

1. **RegionSalesData.cs** コードファイルをダブルクリックして開きます。
2. 次の名前空間をインポートします。

▸ C# コードの書き方

C#

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

3. これらの名前空間の下で、コードファイルの形式を設定します。

▶ C# コードの書き方

C#

```
namespace RegionSales
{
    public class RegionSalesData
    {
    }

    }
}
```

4. 次のサンプルのように RegionSalesData クラスを編集します。

▶ C# コードの書き方

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; }
    }
```

この手順では、RegionSalesData コードファイルにコードを追加しました。次の手順では、このアプリケーションを実行します。

手順4：アプリケーションの実行

この手順ではアプリケーションを実行します。

[F5]キーを押すか、デバッグを開始して、アプリケーションを実行します。次の図のようになります。

それぞれのタイプの C1Sparkline がアプリケーションに表示されます。

タスク別ヘルプ

タスク別ヘルプは、ユーザーの皆様が Visual Studio でのプログラミングに精通しており、C1Sparkline コントロールの一般的な使用方法を理解していることを前提としています。**Sparkline for UWP** 製品を初めて使用される場合は、まず「**クイックスタート**」を参照してください。

このセクションの各トピックは、**Sparkline for UWP** 製品を使用して特定のタスクを実行するための方法を提供します。

また、タスク別ヘルプトピックは、新しい Windows ストアアプリケーションが既に作成されていることを前提としています。

X 軸の表示

C1Sparkline コントロールの X 軸を簡単に表示することができます。

XAML の場合

XAML マークアップを使用して X 軸を表示するには、次のコードを <c1:C1Sparkline> タグに追加します。

```
DisplayXAxis = "True"
```

コードの場合

コードを使用して X 軸を表示するには、次のコードを **InitializeComponent()** メソッドに追加します。

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

DateAxis の使用

C1Sparkline データが一定期間にわたって表示されるように、X 軸の書式を設定することができます。

コードの場合

次のコードを使用して、C1Sparkline の **DateAxisData** プロパティを設定します。

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

C1Sparkline の外観の変更

C1Sparkline コントロールには固有の外観プロパティがあります。これらのプロパティを使用して、アプリケーションをカスタマイズすることができます。外観プロパティの詳細については、「**C1Sparkline の外観のプロパティ**」トピックを参照してください。

C1Sparkline の外観プロパティは XAML マークアップまたはコードで設定できます。

XAML の場合

次のマークアップを <c1:C1Sparkline> タグに挿入して、コントロールの外観プロパティをいくつか設定します。

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

コードの場合

C1Sparkline の外観プロパティを設定するには、コードを次のように設定します。

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