Sparklines for WinForms

2018.11.26 更新

グレープシティ株式会社

Sparklines for WinForms

目次

Sparklines for WinForms Edition

Sparkline for WinForms, 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, legend, and title. Sparklines are at times even more informative than any of usual chart types, just because of its simplicity. A sparkline is positioned near to its data for the greatest impact.

The Sparkline control allows you to create sparklines which adds rich visualization capability to your data without taking too much space. Sparkline control can be used as a standalone control or as a nested control in other container controls such as dashboard.

To work with Sparkline control, you need to create an instance of **C1Sparkline** class.

クイックスタート

The topic describes how to add Sparkline control to your Windows Form application. This topic comprises following three steps:

- **Step 1: Add a C1Sparkline control**
- **Step 2: Create a Data Source**
- **Step 3: View C1Sparkline control**

The following image shows how the Sparkline control appears after completing the above steps.

Step 1: Add a C1Sparkline control

1. Create a new Windows Forms application.
2. Set the height and width of the form to 350
3. Drag and drop the **C1Sparkline** control from the Toolbox onto your from.
4. Set the height and width of the sparkline control to 250.

Step 2: Create a Data Source

In this step, you add a class to your project 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.

C#

```
class SampleData
    {
        public List<double> Sales
        {
            get
            {
                List<double> data = new List<double>() { 1.0, -1.0, 2.0, -3.0,
4.0, 5.0, -5.0, 2.0 };
                return data;
```

```
            }
        }
    }
```

Step 3: View C1Sparkline control

The **SampleData.cs** class added in the above step returns a collection of numeric values using the '**Sales**' property defined in the class. 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 instance of the SampleData class and assign the value returned by the Sales property to **Data** property of Sparkline control.

1. In the Form1_Load event, create an instance of SampleData class and assign the value returned by the **Sales** property to the control's **Data** property.

C#

```
SampleData sampleData = new SampleData();
c1Sparkline1.Data = sampleData.Sales;
```

Click **Build | Build Solution** to build the project. Run the application and observe how the **C1Sparkline** control appears at runtime.

コアクラス

The following table lists some of the key classes and their properties. YOu can click on the cross-references to get to the API documentation that provides detailed description for the key members.

C1Sparkline
Properties: **Data**, **SparklineType**, **DispayXAxis**, **ShowFirst**, **ShowLast**, **ShowHigh**, **ShowLow**, **ShowNegative**, **ShowMarkers**
SparklineTheme

主な特長

As a lightweight data visualization control, Sparkline provides various features that makes it suitable for depicting trend lines and variation curves. The main features of the Sparkline control are as follows.

- **Lightweight and Quick Visualization**
  The Sparkline control allows you to quickly visualize data in a meaningful way. It also helps you to analyze data trends very easily.
- **Sparkline Types**
  The Sparkline control supports three different types of sparklines to cater diverse business needs, including Column, Line, and WinLoss.
- **Markers**
  Sparkline allows you to highlight data points, in order to make the sparkline more readable. Moreover, the markers can be tailored to suit user requirements.
- **Axis**
  Sparkline allows you to display the 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.
- **Styling**
  The Sparkline control provides numerous styling options to customize its appearance such as the color of axis, data points (first, last, high, low and negative), and the complete series.
- **Integration with Controls**
  It is very common for sparklines to be added to any of the data management controls such as grids or dashboard layout as they are compact yet extremely effective in data visualization. The Sparkline control can be easily integrated with controls like FlexGrid and DashboardLayout.
- **Data Binding**
  The Sparkline control can easily bind to any enumerable collection of data values i.e. to any class that implements the IEnumerable interface. Also, the class can implement the INotifyCollectionChanged interface to have support for modifying data after binding.

Work with Sparkline Control

Learn about concepts that help you to understand how best to use a TreeMap control.

This section contains information about

Data Binding
: Learn how to implement data binding in the Sparkline control.

Sparkline Types
: Learn about the Sparkline types; Line, Column, and WinLoss.

Markers
: Learn how work with markers in the Sparkline control.

Axis
: Learn how to display an x-axis in the Sparkline control.

Styling
: Learn how to work with styles in the Sparkline control.

データ連結

Data binding is the core for any data visualization control. The Sparkline control can easily bind to any enumerable collection of data values i.e. to any class that implements the **IEnumerable** interface. Also, the class can implement the **INotifyCollectionChanged** interface to have support for modifying data after binding.

The topic describes how to add Sparkline control to your Windows Form application. This topic comprises following three steps:

- **Step 1: Add a C1Sparkline control**
- **Step 2: Create a Data Source**
- **Step 3: View C1Sparkline control**

The following image shows how the Sparkline control appears after completing the above steps.

Step 1: Add a C1Sparkline control

1. Create a new Windows Forms application.
2. Set the height and width of the form to 350
3. Drag and drop the **C1Sparkline** control from the Toolbox onto your from.

4. Set the height and width of the sparkline control to 250.

Step 2: Create a Data Source

In this step, you add a class to your project 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.

C#

```
class SampleData
    {
        public List<double> Sales
        {
            get
            {
                List<double> data = new List<double>() { 1.0, -1.0, 2.0, -3.0,
4.0, 5.0, -5.0, 2.0 };
                return data;
            }
        }
    }
```

Step 3: View C1Sparkline control

The **SampleData.cs** class added in the above step returns a collection of numeric values using the '**Sales**' property defined in the class. 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 instance of the SampleData class and assign the value returned by the Sales property to **Data** property of Sparkline control.

1. In the Form1_Load event, create an instance of SampleData class and assign the value returned by the **Sales** property to the control's **Data** property.

C#

```
SampleData sampleData = new SampleData();
c1Sparkline1.Data = sampleData.Sales;
```

Click **Build | Build Solution** to build the project. Run the application and observe how the **C1Sparkline** control appears at runtime.

Sparklineの種類

The Sparkline control supports three different sparkline 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.

The different sparkline types are explained in greater detail below:

- **Line**

  A line sparkline consists of data points connected by line segments. It is best suited for visualizing continuous

data and can be used to visualize sales figures, stock values or website traffic.

By default, Sparkline renders as a line sparkline.

- **Column**

  Each data point in this type of sparkline is depicted as a vertical rectangle/column. In a column sparkline, positive data points are drawn in upward direction, while negative data points are drawn in downward direction. Column sparklines are used to facilitate comparison and are best suited for visualizing categorial data, for example, to visualize the revenues/profits earned from different departments in a store.

- **WinLoss**

  WinLoss sparkline displays data points through equal-sized columns drawn in upward and downward direction. The winloss sparkline is only concerned with whether a datapoint is positive or negative, it does not take the relative size of the data point into account. It is used to visualize a win/loss scenario. Columns drawn in upward direction indicate a win, while downward columns indicate a loss. For example, winloss sparkline can be used to track a sports season.

The **C1Sparkline** class provides **SparklineType** property to set the sparkline type in designer or code. The SparklineType property accepts the following values from the SparklineType enumeration:

1. **Line** - Allows you to draw the line sparkline.
2. **Column** - Allows you to draw the column sparkline.
3. **WinLoss** - Allows you to draw the winloss sparkline.

The code example below shows how you can set the SparklineType property to a specific sparkline type.

C#

```
//Setting the Sparkline type
sparkline.SparklineType = SparklineType.Column;
```

マーカー

Markers are symbols that are used to highlight or emphasize certain data points. With the help of markers, the sparkline becomes more readable and it becomes easier to distinguish between specific data points, such as high or low values.

The Sparkline control supports markers only for the line sparkline type. In a line sparkline, markers are represented by the help of dot symbols which are shown on the data points. You can show all the data points in a line sparkline by setting the **ShowMarkers** property provided by the **C1Sparkline** class to **true**. This highlights all the data points in brown color.

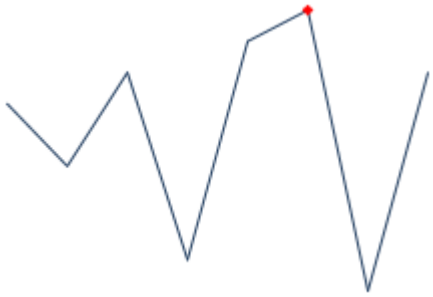
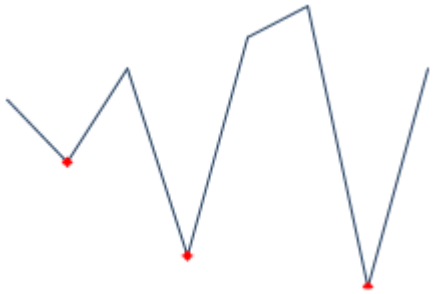
The following example code shows how you can set the ShowMarkers property.

C#

```
//Highlights all the data points
sparkline.ShowMarkers = true;
```

The **ShowMarkers** 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.

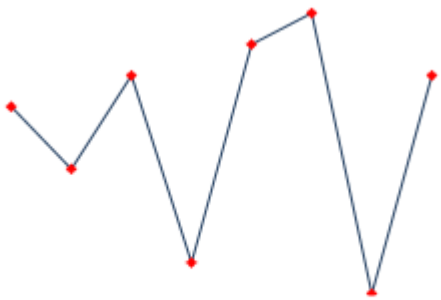
Property	Description	Output
ShowFirst	You can set this property to true in code to highlight the first data point in the output as shown in the image alongside. By default, the marker highlights the first data point in maroon color.	
ShowLast	You can set this property to true in code to highlight the 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	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	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	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.	

Customizing the color of Data Points

As mentioned above, various data points are rendered in a default color. However, Sparkline allows you to modify this default behavior and highlight data points in a color of your choice, making Sparkline more customizable for users from appearance perspective.

The following image shows how a Sparkline control appears after setting the MarkersColor property to Red.



Markers highlighting data points in Red Color

The following code example shows how to customize the markers in the Sparkline control.

C#

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

Following is the list of properties that can be used to set the colors for highlighting specific data points.

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

軸

The Sparkline control supports an x-axis, which can either be displayed or kept hidden as per the user requirements. By default, the Sparkline control renders without any axis. However, at times, there may be a need to display the horizontal axis, to better visualize values greater than and less than zero. This is primarily helpful with line sparkline type to differentiate between the negative and the positive data points.

<table>
<tr>
<td>  </td>
<td>  </td>
</tr>
<tr>
<td>

```
//Displays the horizontal axis

sparkline.DisplayXAxis = true;
```

</td>
<td>

```
//Displays the horizontal axis
sparkline.DisplayXAxis = false;
```

</td>
</tr>
</table>

スタイル設定

We can customize the appearance of the Sparkline control, such as the color of the axis, color of the data points and color of the series, by using the Styles property provided by the C1Sparkline class, which returns an object of the SparklineTheme class.

For customizing the color of the horizontal axis and the series, you can use the **AxisColor** and **SeriesColor** properties of the SparklineTheme class. Additionally, you can change the distance between two bars of a column/winloss

sparkline with the help of the **BarDistance** property. The **LineWeight** property can be used to change the thickness of the sparkline lines in a line sparkline by specifying the desired line weight in points.

The following image shows how Sparkline appears after appyling styles.

The following code example shows how you can apply styles to the Sparkline control.

C#

```
//Sets the color of the axis
c1Sparkline1.Styles.AxisColor = Color.Orange;
//Sets the distance between two bars of a bar sparkline
c1Sparkline1.Styles.BarDistance = 30;
//Sets the color of the sparkline
c1Sparkline1.Styles.SeriesColor = Color.LightPink;
//Sets the line weight
c1Sparkline1.Styles.LineWeight = 2;
//Sets the color of the first data point
c1Sparkline1.Styles.FirstMarkerColor = Color.Yellow;
```

チュートリアル

Microsoft DataGridViewでSparklineの使用

This walkthrough explains the steps to show sparklines in **Microsoft DataGridView** control. The sparkline control cannot be directly added to the DataGridView cells but alternatively you can render it as an image in the DataGridView cells. To achieve the same, you first need to add an unbound DataGridViewImageColumn to the grid and then use the CellPainting event to render the Sparkline control as an image in the added unbound column. The walkthorugh topic includes the following three steps.

- **Step 1: Add an MS DataGridView control**
- **Step 2: Create a Data Source**
- **Step 3: Show Sparklines in DataGridViewImageColumn**

The following image shows the MS DataGridView control representing the C1Sparkline control in DataGridViewImageColumn.

Product	Jan'18	Feb'18	Mar'18	Quaterly Sales Trend
Chang	136.94	991.48	67.08	
Tofu	244.8	483.3	864.13	
Filo Mix	497.79	868.31	406.33	
Chocolade	268.65	571.5	687.81	
Ikura	464.57	864.17	656.59	
Pavlova	106.22	382.05	986.12	
Konbu	634.33	484.9	871.87	
Ipoh Coffee	95.18	512.35	674.23	

Step 1: Add an MS DataGridView control

1. Create a new Windows Forms application. For more information on how to create an WinForms application, see Creating a Windows Forms Project topic.
2. Drag and drop **DataGridView** control from the Toolbox onto your form.
3. From the **Properties** window, set its **Dock** property to **Fill**.

Step 2: Create a Data Source

1. Create a **DataTable** using the following method '**InitDataTable'**. This DataTable will be used as the **DataSource** for the DataGridView.

Form1.cs

```
private void InitDataTable() {
 _dataTable = new DataTable();
 _dataTable.Columns.Add("Product", typeof(String));
 _dataTable.Columns.Add("Jan'18", typeof(double));
 _dataTable.Columns.Add("Feb'18", typeof(double));
 _dataTable.Columns.Add("Mar'18", typeof(double));
 string[] productNames = {
  "Chang",
  "Tofu",
  "Filo Mix",
  "Chocolade",
```

```
  "Ikura",
  "Pavlova",
  "Konbu",
  "Ipoh Coffee"
 };
 Random random = new Random();
 for (int i = 0; i < productNames.Length; i++) {
  _dataTable.Rows.Add(new object[] {
   productNames[i], Math.Round(random.NextDouble() * 1000, 2),
Math.Round(random.NextDouble() * 1000, 2), Math.Round(random.NextDouble() *
1000, 2)
  });
 }}
```

Step 3: Show Sparklines in DataGridViewImageColumn

1. Create a method named **SetUpGrid** to add an unbound DataGridViewImageColumn to the **DataGridView** control. This column is used for showing the sparkline as an image in the DataGridViewImageColumn.

   Also subscribe to the CellPainting event. This event is used to render the sparklines in the grid initially. It is also used to re-render the sparkline at the time of resizing, sorting etc when the cell displaying the sparkline needs to be repainted.

   Form1.cs

```
private void SetUpGrid() {
 dataGridView1.DataSource = _dataTable;
 //Add an unbound DataGridViewImageColumn to the DataGridView. This column will
be used to show the sparkline as an image
 DataGridViewImageColumn dataGridViewImageColumn = new
DataGridViewImageColumn();
 dataGridViewImageColumn.Name = "Quaterly Sales Trend";
 dataGridViewImageColumn.HeaderText = "Quaterly Sales Trend";
 dataGridView1.Columns.Add(dataGridViewImageColumn);
 //Setting additional properties of the DataGridView (optional)
 dataGridView1.Font = new Font(FontFamily.GenericSansSerif, 9.5 f,
FontStyle.Regular);
 dataGridView1.RowHeadersVisible = false;
 dataGridView1.AllowUserToAddRows = false;
 dataGridView1.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;
 dataGridView1.ColumnHeadersDefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;
 //Subscribe to the DataGridView's CellPainting event
 dataGridView1.CellPainting += DataGridView1_CellPainting;
}
```

2. To render the Sparkline control as an image in the DataGridViewImageColumn, use the DataGridView's **CellPainting** event. In the CellPainting event handler, first create an image of the Sparkline control with the help of Control.DrawToBitmap method. Then, render the created image in the DataGridViewImageColumn cells using the Graphics.DrawImage method as shown below.

   Form1.cs

```
private void DataGridView1_CellPainting(object sender,
DataGridViewCellPaintingEventArgs e) {
 if (e.RowIndex >= 0 && e.RowIndex < dataGridView1.Rows.Count && e.ColumnIndex
== 4) {
  e.Handled = true;
  DataRow row = _dataTable.DefaultView.ToTable().Rows[e.RowIndex];
  //Initialize the Sparkline control
  C1Sparkline sparkline = new C1Sparkline();
  sparkline.Width = e.CellBounds.Width;
  sparkline.Height = e.CellBounds.Height;
  sparkline.BackColor = Color.White;
  sparkline.Styles.LineWeight = 1.6;
  //Add a marker to highlight the data point with the highest value.
  sparkline.ShowHigh = true;
  //Specify the Sparkline data
  List<double> values = new List < double > ();
  for (int i = 1; i <= 3; i++) {
   values.Add(Convert.ToDouble(row.ItemArray[i]));
  }
  sparkline.Data = values;
  //Render the sparkline to a bitmap
  Bitmap bitmap = new Bitmap(e.CellBounds.Width, e.CellBounds.Height);
  sparkline.DrawToBitmap(bitmap, sparkline.Bounds);
  //Draws the Sparkline image in the DataGridView cell
  e.Graphics.DrawImage(bitmap, new Point(e.CellBounds.X, e.CellBounds.Y));
  e.Paint(e.CellBounds, DataGridViewPaintParts.Border);
 }
}
```

3. Finally, call the **InitDataTable** and **SetUpGrid** methods in the **Form1_Load** event handler.

Form1.cs

```
private void Form1_Load(object sender, EventArgs e) {
//Creates DataTable which will be used as the DataSource for the DataGridView
InitDataTable();
//Adds an unbound column to the DataGridView and sets the DataGridView
properties
SetUpGrid();
}
```

Run the application. Observe that the sparkline control gets displayed in the '**Quarterly Sales Trend**' column of the MS DataGridView control.