对于WPF版Chart的重点提示

针对WPF以及Silverlight版Chart，以下几个常用提示将在您使用C1Chart控件时对您有所帮助。
提示技巧一：使用提示技巧一：使用BeginUpdate()/EndUpdate()方法以提高性能方法以提高性能
这将极大的提高性能，因为重绘动作仅仅在调用了EndUpdate()方法之后进行一次。
例如：

Visual Basic

```
' 开始批量更新
C1Chart1.BeginUpdate()
Dim nser As Integer = 10, npts As Integer = 100
For iser As Integer = 1 To nser
' 创建数据数组
Dim x(npts - 1) As Double, y(npts - 1) As Double For ipt As Integer = 0 To npts - 1 x(ipt) = ipt
y(ipt) = (1 + 0.05 * iser) * Math.Sin(0.1 * ipt + 0.1 * iser)
Next
' 创建数据系列
Dim ds = New XYDataSeries() ds.XValuesSource = x ds.ValuesSource = y C1Chart1.Data.Children.Add(ds)
Next
' 设置图表类型
C1Chart1.ChartType = ChartType.Line
' 结束批量更新
C1Chart1.EndUpdate()
```

C#

```
// 开始批量更新
c1Chart1.BeginUpdate();
```

```
int nser = 10, npts = 100; for (int iser = 0; iser < nser; iser++)
{
// 创建数据数组
<ac:structured-macro ac:name="unmigrated-wiki-markup" ac:schema-version="1"
ac:macro-id="d8e47b9f-23a2-4bff-924b-11c44b1a6c74"><ac:plain-text-body><![CDATA[ double[] x = new double[npts], y = new do
uble[npts]; for (int ipt = 0; ipt < npts; ipt++)
]]></ac:plain-text-body></ac:structured-macro>
<ac:structured-macro ac:name="unmigrated-wiki-markup" ac:schema-version="1"
ac:macro-id="686af20c-ccb6-40e2-9deb-ab60e16a88be"><ac:plain-text-body><![CDATA[ { x[ipt] = ipt; y[ipt] = (1 + 0.05 *
iser) * Math.Sin(0.1 * ipt + 0.1 * iser); }
]]></ac:plain-text-body></ac:structured-macro>
// 创建数据系列
XYDataSeries ds = new XYDataSeries(); ds.XValuesSource = x; ds.ValuesSource = y; c1Chart1.Data.Children.Add(ds); }
// 设置图表类型
c1Chart1.ChartType = ChartType.Line;
// 结束批量更新
c1Chart1.EndUpdate();
```

提示技巧二：对于大量的数据数组，使用折线图或者面积图提示技巧二：对于大量的数据数组，使用折线图或者面积图
当您需要展示大量数据值时，折线图和面积图提供了最优的展示性能。
为了获取更佳的性能，通过设置attached属性，LineAreaOptions.OptimizationRadius启用大量数据展示优化。例如：

Visual Basic

```
LineAreaOptions.SetOptimizationRadius(C1Chart1, 1.0)
```

C#

```
LineAreaOptions.SetOptimizationRadius(c1Chart1, 1.0);
```

强烈建议您使用比较小的值作为范围，比如说1.0~2.0。一个较大的值将影响绘图的精度。
提示技巧三：使用提示技巧三：使用DataSeries.PlotElementLoaded事件更新绘图区元素的外观和行为事件更新绘图区元素的外观和行为
当任意绘图区元素（条形，柱形或者饼图，等等）加载完成时，将出发PlotElementLoaded（在线文档'PlotElementLoaded事件'）事件。在此事件中，您可以访问此绘图区元素，以及其关联的数据点。
以下代码按照y-轴的值的不同设置数据点的颜色。例如：

Visual Basic

```
' 创建数据数组
Dim npts As Integer = 100
Dim x(npts - 1) As Double, y(npts - 1) As Double For ipt As Integer = 0 To npts - 1 x(ipt) = ipt

y(ipt) = Math.Sin(0.1 * ipt)
Next
```

```
' 创建数据系列
Dim ds = New XYDataSeries() ds.XValuesSource = x ds.ValuesSource = y
'设置事件处理器
AddHandler ds.PlotElementLoaded, AddressOf PlotElement_Loaded
' 添加数据系列至图表
C1Chart1.Data.Children.Add(ds)
' 设置图表类型
C1Chart1.ChartType = ChartType.LineSymbols ...
' event handler
Sub PlotElement_Loaded(ByVal sender As Object, ByVal args As EventArgs)
Dim pe = CType(sender, PlotElement)
If Not TypeOf pe Is Lines Then
Dim dp As DataPoint = pe.DataPoint ' 正规化 y-值（范围从 0 到 1）
Dim nval As Double = 0.5 * (dp.Value + 1)
' 填充色从蓝色（－1）变化到红色（＋1）
pe.Fill = New SolidColorBrush(Color.FromRgb(CByte(255 * nval), _ 0, CByte(255 * (1 - nval))))
End If
End Sub
```

C#

```
// 创建数据数组
<ac:structured-macro ac:name="unmigrated-wiki-markup" ac:schema-version="1"
ac:macro-id="043b5e9e-7d27-454c-bcbe-33b20e7f9030"><ac:plain-text-body><![CDATA[ int npts = 100; double[] x = new double[n
pts], y = new double[npts]; for (int ipt = 0; ipt < npts; ipt++)
]]></ac:plain-text-body></ac:structured-macro>
<ac:structured-macro ac:name="unmigrated-wiki-markup" ac:schema-version="1"
ac:macro-id="a2063254-2f1b-4d0a-ae20-bbb4e1ddd72a"><ac:plain-text-body><![CDATA[ { x[ipt] = ipt; y[ipt] = Math.Sin(0.1 *
ipt); }
]]></ac:plain-text-body></ac:structured-macro>
// 创建数据系列
XYDataSeries ds = new XYDataSeries(); ds.XValuesSource = x; ds.ValuesSource = y;
//设置事件处理器
ds.PlotElementLoaded += (s, e) => { PlotElement pe = (PlotElement)s; if (!(pe is Lines)) // skip lines
{
DataPoint dp = pe.DataPoint;

// 正规化 y-值（范围从 0 到 1）
double nval = 0.5*(dp.Value + 1);
// 填充色从蓝色（－1）变化到红色（＋1）
pe.Fill = new SolidColorBrush(
Color.FromRgb((byte)(255 * nval), 0, (byte)(255 * (1-nval))));
}
};
// 添加数据系列至图表
c1Chart1.Data.Children.Add(ds);
// 设置图表类型
c1Chart1.ChartType = ChartType.LineSymbols;
```

提示技巧四：数据点标签及工具提示提示技巧四：数据点标签及工具提示为创建一个数据点标签或工具提示，您应当设置模版至PointLabelTemplate或者PointToolTipTemplate属性。以下示例代码将演示如何显示每一个数据点的索引。

XAML

```
<c1chart:C1Chart Name="c1Chart1" ChartType="XYPlot">
<c1chart:C1Chart.Data>
<c1chart:ChartData>
<!-- source collection -->
<c1chart:ChartData.ItemsSource>
<PointCollection>
<Point X="1" Y="1" />
<Point X="2" Y="2" />
<Point X="3" Y="3" />
<Point X="4" Y="2" />
<Point X="5" Y="1" />
</PointCollection>
</c1chart:ChartData.ItemsSource>
<c1chart:XYDataSeries SymbolSize="16,16"
XValueBinding="{Binding X}" ValueBinding="{Binding Y}">
<c1chart:XYDataSeries.PointLabelTemplate>
<DataTemplate>
<!-- display point index at the center of point symbol -->
<TextBlock c1chart:PlotElement.LabelAlignment="MiddleCenter"

Text="{Binding PointIndex}" />
</DataTemplate>
</c1chart:XYDataSeries.PointLabelTemplate>
</c1chart:XYDataSeries>
</c1chart:ChartData>
</c1chart:C1Chart.Data>
</c1chart:C1Chart>
```

提示技巧五：将图表保存为图片提示技巧五：将图表保存为图片以下方法将一个图表的图片保存为Png文件格式。

Visual Basic

```
Sub Using stm = System.IO.File.Create(fileName) c1Chart1.SaveImage(stm, ImageFormat.Png)

End Using
```

C#

```
using (var stm = System.IO.File.Create(fileName))

{ c1Chart1.SaveImage(stm, ImageFormat.Png);
}
```

提示技巧六：打印图表提示技巧六：打印图表
以下代码在默认的打印机上采用默认的设置打印指定的图表。例如：

Visual Basic

```
Dim pd = New PrintDialog() pd.PrintVisual(C1Chart1, "chart")
```

C#

```
new PrintDialog().PrintVisual(c1Chart1, "chart");
```

提示技巧七：混合直角坐标图表类型提示技巧七：混合直角坐标图表类型
您可以容易地在同一个直角坐标系的绘图区通过使用ChartType属性混合显示不同的图表类型。
以下代码创建了三个数据系列，第一个是面积图，第二个是step图，第三个为默认的图表类型（折线图）。

Visual Basic

```
Dim nser As Integer = 3, npts As Integer = 25
For iser As Integer = 1 To nser
' 创建数据数组
Dim x(npts - 1) As Double, y(npts - 1) As Double For ipt As Integer = 0 To npts - 1 x(ipt) = ipt
y(ipt) = (1 + 0.05 * iser) * Math.Sin(0.1 * ipt + 0.1 * iser)
Next
' 创建数据系列
Dim ds = New XYDataSeries() ds.XValuesSource = x ds.ValuesSource = y C1Chart1.Data.Children.Add(ds)

Next
'设置图表类型
C1Chart1.ChartType = ChartType.Line
```

```
' 第一系列
C1Chart1.Data.Children(0).ChartType = ChartType.Area
' 第二系列
C1Chart1.Data.Children(1).ChartType = ChartType.Step
```

C#

```
int nser = 3, npts = 25; for (int iser = 0; iser < nser; iser++)
{
// 创建数据数组
<ac:structured-macro ac:name="unmigrated-wiki-markup" ac:schema-version="1"
ac:macro-id="87f4bc89-bc83-49d6-a9a1-81792c909300"><ac:plain-text-body><![CDATA[
double[] x = new double[npts], y = new double[npts]; for (int ipt = 0; ipt <
npts; ipt++)
]]></ac:plain-text-body></ac:structured-macro>
<ac:structured-macro ac:name="unmigrated-wiki-markup" ac:schema-version="1"
ac:macro-id="7b7e2d1b-0aff-4848-bcdb-5e704035bb80"><ac:plain-text-body><![CDATA[
{ x[ipt] = ipt; y[ipt] = (1 + 0.05 * iser) * Math.Sin(0.1 * ipt + 0.1 * iser);
}
]]></ac:plain-text-body></ac:structured-macro>
// 创建数据系列
XYDataSeries ds = new XYDataSeries(); ds.XValuesSource = x; ds.ValuesSource =
y; c1Chart1.Data.Children.Add(ds); }
//设置图表类型
c1Chart1.ChartType = ChartType.Line;
// 第一系列
<ac:structured-macro ac:name="unmigrated-wiki-markup" ac:schema-version="1"
ac:macro-id="712b46c0-85bd-4264-b003-c04b65953b49"><ac:plain-text-body><![CDATA[
c1Chart1.Data.Children[0].ChartType = ChartType.Area;
]]></ac:plain-text-body></ac:structured-macro>
// 第二个系列
<ac:structured-macro ac:name="unmigrated-wiki-markup" ac:schema-version="1"
ac:macro-id="2678bb1d-0f0a-4b87-bc55-341f6d80f7e0"><ac:plain-text-body><![CDATA[
c1Chart1.Data.Children[1].ChartType = ChartType.Step;
```

```
]]></ac:plain-text-body></ac:structured-ma
```