WPF及Silverlight版本Chart

WPF及Silverlight版Chart通过强大的渲染系统，丰富的带有样式的元素，动画以及数据绑定能力给图表的表示带来革命性的变化。
在WPF中，您可以在一个智能客户端应用程序或者利用XBAP支持的强项将程序部署到Web平台。
通过使用内置的主题，图表类型以及多种多样的调色板，您定制化一个图表的过程只需要轻点鼠标即可完成，完全不需要额外的编码工作。仅需一步设置，就可以转换您的数据并添加专业级的图表至您的应用程序；WPF及Silverlight版
Chart支持全部的热门图表类型，包括条形图，柱状图，折线图，面积图，饼图以及更多其他图表类型。今天，把你的数据可视化更上一层楼！
入门接下来带您开始了解和学习Chart控件。

主要特性

WPF及Silverlight版Chart包含以下独有功能：
超过超过40种的图表类型种的图表类型
可以为您的Silverlight应用程序从30种常用的2D图表类型选择您所需要的图表展示。可用的图表类型包括折线图，散点图，饼图，面积图以及更多其他种类的图表。
仅需设置两个属性就可以得到专业的设计仅需设置两个属性就可以得到专业的设计
图表包含12种内置的主题以及22种内置的调色板。仅需在Visual
Studio中设置一个属性即可轻松完成设置！主题将应用到整个图表区域，而调色板仅应用于图表元素（条形，点，饼图块，等等）。将主题和不同的调色板组合，可以毫不费力地创建无尽的外观组合。
标签以及工具提示标签以及工具提示
可以通过标签或者工具提示形式在图表元素上显示相关联的数据的值。任意的UI元素可以用作标签和工具提示，并允许完全定制。
图表图例图表图例
仅需设置一个属性，即可通过C1ChartLegend控件创建一个关联到图表的独立的图表图例。该设计在排布图例区以及设置图例区样式时，提供了最大限度的灵活性。
完全可交互式图表完全可交互式图表
通过允许放大/缩小，拉伸以及滚动图表，极大的提高了最终用户所获得的用户体验。
多重坐标轴多重坐标轴
图表支持多重，相关联的坐标轴，只需要简单地定义一个Axis对象并添加到图表的View.Axes集合，即可将这些坐标轴添加到图表。
对数轴刻度对数轴刻度
图表支持任意底数的对数坐标轴刻度。
趋势线趋势线
可以通过趋势线分析图表中的数据走势。图表支持几种不同的自动趋势线，包括多项式，指数，对数，乘幂，傅里叶，平均值，最小值以及最大值。
高亮和阴影效果高亮和阴影效果
创建具有高亮效果的边框，并可以在绘图区元素之后添加不同柔和程度的阴影。
堆叠图表堆叠图表堆叠图表为展示复杂数据提供了一种简单的方式。折线图，面积图，条形图，雷达图以及地块图可以堆叠在一起以便在有限的空间内显示更加复杂的数据。
动态图形动态图形
图表采用了Silverlight平台下可用的动态图形的优势，包括透视和动画。
支持支持Silverlight Toolkit 主题（仅主题（仅Silverlight））
除了12种内置的主题之外，图表还附带了最流行的Microsoft Silverlight Toolkit主题，包括ExpressionDark，
ExpressionLight，WhistlerBlue，RainerOrange，ShinyBlue，以及BureauBlack。
XBAP支持（仅支持（仅WPF））
在WPF版本中，C1Chart完全兼容XBAP发布功能。XBAP发布方式允许将全部功能的应用程序发布到支持的客户端浏览器，无须使用Windows安装程序。

快速入门

请参见以下步骤：

第一步：添加一个图表至您的工程

在此步骤中，您将开始使用WPF及Silverlight版Chart在Visual Studio 中创建一个图表应用。当您添加C1Chart（在线文档'C1Chart
类'）控件至您的Visual Studio工程，您将看到一个具有功能的带有假数据的柱状图。完成以下步骤：

1. 在 Visual Studio 中创建一个新的 WPF 或者 Silverlight 应用程序。
   a. 选择 File | New | Project。将显示新建工程对话框。
   b. 在此新建工程对话框中，在位于左侧的面板中选择一种语言，并选择一个模版。
      i. 对于WPF应用程序，选择 Windows Desktop。接下来在中间的面板中选择WPF应用程序。
      ii. 对于Silverlight程序，选择Silverlight。接下来从中间的面板中选择 Silverlight 应用程序。
      iii. 为您的应用程序命名，并选择OK。您的应用程序将创建并打开。
2. 通过右键单击 Reference 文件夹并选择 Add Reference，向您的应用程序添加引用。将打开 Reference

Manager。

1. a. 浏览并定位到适合您工程的程序集引用。
      i. 对于WPF应用程序，添加以下引用：C1.WPF.Chart.4.dll
      ii. 对于 Silverlight 应用程序，添加以下引用：C1.Silverlight.Chart.5.dll
2. 在您的 <Window> 或者 <UserControl>
   标签中添加以下命名空间声明。如果您在操作XAML标记语言，这将在接下来允许您在该工程中添加图表数据。

   ```
   XAML
   xmlns:System="clr-namespace:System;assembly=mscorlib"
   ```

3. 将光标移动到位于Window或者UserControl内部的<Grid></Grid>标签之间。具体是Window还是UserControl取决于您所创建的应用程序类型。
4. 在Visual Studio 的工具栏中找到 C1Chart
   控件。双击该控件以将其添加至您的应用程序。XAML标记语言将重新生成如下所示的代码：

   ```
   XAML
   ```

```
<Grid x:Name="LayoutRoot">
<c1chart:C1Chart></c1chart:C1Chart>

</Grid>
```

5. 为您的图表命名，之后您可以通过代码对其进行访问。您的标记语言将类似于以下的代码示例：

XAML

```
<c1chart:C1Chart Margin="0,0,8,8" MinHeight="160" MinWidth="240"

Name="c1Chart1">
</c1chart:C1Chart>
```

您所完成的部分您所完成的部分
到此您已经成功地创建了包含一个 C1Chart 控件的 WPF 或者 Silverlight 应用程序。在下一步（第二步：向图表添加数据）中，您将为 C1Chart 添加数据。
第二步：向图表添加数据
在上一步中，您完成了向窗体添加添加了C1Chart（在线文档'C1Chart 类'）控件的工作。在这一步，您将为其添加一个 DataSeries（在线文档'DataSeries 类'）对象以及数据。有两种方式添加一个DataSeries：通过XAML标记语言或者通过代码进行添加。下面的选项卡包含了全部的两种向图表添加数据的方式。选择合适的选项卡并完成其中的步骤：
1. 通过编辑<c1chart:C1Chart>标签设置ChartType，使得结果类似下面的代码：XAML<c1chart:C1Chart Name="c1Chart1" ChartType="Bar" Margin="0,0,8,8" MinHeight="160" MinWidth="240" Content="C1Chart"></c1chart:C1Chart> 2. 在XAML中，您可以通过使用C1ChartData对象添加您的数据：XAMLc1chart:C1Chart Name=<"c1Chart1" ChartType="Bar" Margin="0,0,8,8" MinHeight="160" MinWidth="240" Content="C1Chart">>c1chart:C1Chart.Data< <c1chart:ChartData> <c1chart:ChartData.ItemNames> <x:Array Type="{x:Type System:String}"> <System:String>Hand Mixer</System:String> <System:String>Stand Mixer</System:String> <System:String>Can Opener</System:String> <System:String>Toaster</System:String> <System:String>Blender</System:String> <System:String>Food Processor</System:String> <System:String>Slow Cooker</System:String> <System:String>Microwave</System:String> </x:Array> </c1chart:ChartData.ItemNames> <c1chart:DataSeries Values="80 400 20 60 150 300 130 500" AxisX="Price" AxisY="Kitchen Electronics" Label="Price"/>

```
</c1chart:ChartData>
</c1chart:C1Chart.Data>
</c1chart:C1Chart>
```

在本步骤中，我们使用了一个具有八个X值的DataSeries。我们向ChartData添加了字符串类型的ItemName 以表示每一个数据值的字符串名称。我们使用了一个字符串名称的数组表示ItemNames，
因为其中有个别项目的名称包含空格。我们可以使用System:String命名空间，因为我们曾经在第一步中声明了该命名空间。向您的工程

1. 右键单击MainPage.xaml文件，并选择代码视图。2. 直接添加C1.Silverlight.C1Chart命名空间Visual BasicImports C1.Silverlight.Chart C#using C1.Silverlight.Chart;3. 在Windows1类的构造器中添加以下代码以创建一个条形图：Visual Basic' 清除之前的旧数据c1Chart1.Data.Children.Clear()' 添加数据Dim ProductNames As String() = {"Hand Mixer", "Stand Mixer", "Can Opener", "Toaster", "Blender", "Food Processor", "Slow Cooker", "Microwave"}Dim PriceX As Integer() = {80, 400, 20, 60, 150, 300, 130, 500}' 为产品价格创建单一系列Dim ds1 As New DataSeries()ds1.Label = "Price X"'set price datads1.ValuesSource = PriceX' 添加系列至图表c1Chart1.Data.Children.Add(ds1)' 添加项目名称c1Chart1.Data.ItemNames = ProductNames' 设置图表类型c1Chart1.ChartType = ChartType.Bar C#// 清除之前的旧数据

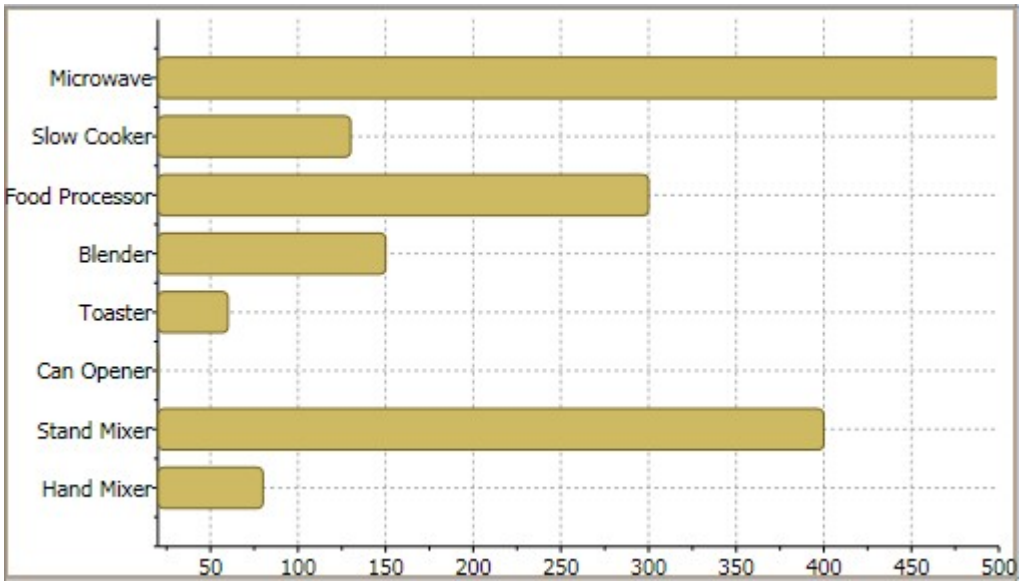
```
c1Chart1.Data.Children.Clear();
// 添加数据
<ac:structured-macro
ac:name="unmigrated-wiki-markup"
ac:schema-version="1"
ac:macro-id="c8d8473b-a120-4554-9cb9-10
add4300032"><ac:plain-text-body><![CDAT
A[ string[] ProductNames = { "Hand
Mixer", "Stand Mixer", "Can Opener", "T
oaster", "Blender", "Food Processor", "
Slow Cooker", "Microwave" }; int[]
PriceX = { 80, 400, 20, 60, 150, 300,
130, 500 };
]]></ac:plain-text-body></ac:structured
-macro>
// 为产品价格创建单一系列
DataSeries ds1 = new DataSeries();
ds1.Label = "Price X";
//设置价格数据
ds1.ValuesSource = PriceX;

// 添加系列至图表
c1Chart1.Data.Children.Add(ds1);

// 添加项目名称
c1Chart1.Data.ItemNames = ProductNames;

// 设置图表类型
c1Chart1.ChartType = ChartType.Bar;
```

您所完成的部分您所完成的部分
至此您已经成功地向C1Chart添加数据，因此当运行您的应用程序时，Y轴将出现字符串值，如下图所示：



第三步：格式化坐标轴

在本步骤中，您将添加一个ChartView（在线文档'ChartView 类'）对象，您可以通过该对象自定义X轴。您可以通过XAML标记语言或者代码完成此步骤。从下方选择适当的选项卡并完成指定步骤：
In XAML

1. 添加ChartView对象，之后您可以设置X-轴以及Y-轴的标题

   ChartView对象表示图表中包含数据（包括坐标轴）的区域。关于图表坐标轴的更多信息，请参见坐标轴。坐标轴标题是UIElement对象，而不仅仅是简单的文本。在该示例中，我们将使用TextBlock元素为X-轴和Y轴的标题指定文本内容。在我们添加了TextBlock元素之后，我们接下来可以做一些格式化工作，比如数修改其前景色并将文本对其设置为居中。

XAML

```
<c1chart:C1Chart >
<c1chart:C1Chart.View>
<c1chart:ChartView>
<c1chart:ChartView.AxisX>
<c1chart:Axis>
<c1chart:Axis.Title>
<TextBlock Text="Price" TextAlignment="Center"
Foreground="Crimson"/>
</c1chart:Axis.Title>
</c1chart:Axis>
</c1chart:ChartView.AxisX>
<c1chart:ChartView.AxisY>
<c1chart:Axis>
<c1chart:Axis.Title>
<TextBlock Text="Kitchen Electronics" TextAlignment="Center" Foreground="Crimson"/>
</c1chart:Axis.Title>
</c1chart:Axis>
</c1chart:ChartView.AxisY>
</c1chart:ChartView>
</c1chart:C1Chart.View>
</c1chart:C1Chart>
```

1. 设置X-轴的值从零开始，并将默认的Axis.MajorUnit单位值从50修改为20。同时设置AutoMin属性的值为

   False，因此我们可以让值从零开始计算的设置生效，否则坐标轴仍然会使用最小的数据值。至此，针对View对象的XAML代码应当如下面所示：| | |

	XAML	
	<c1chart:C1Chart.View> <c1chart:ChartView> <c1chart:ChartView.AxisX> <c1chart:Axis Min="0" Max="500" MajorUnit="20" AutoMin="False"> <c1chart:Axis.Title> <TextBlock Text="Price" TextAlignment="Center" Foreground="Crimson" /> </c1chart:Axis.Title> </c1chart:Axis> </c1chart:ChartView.AxisX> <c1chart:ChartView.AxisY> <c1chart:Axis>	

3. 4.	<c1chart:Axis.Title> <TextBlock Text="Kitchen Electronics" TextAlignment="Center" Foreground="Crimson" /> </c1chart:Axis.Title> </c1chart:Axis> </c1chart:ChartView.AxisY> </c1chart:ChartView> </c1chart:C1Chart.View>	
	在ChartView.AxisX对象的<c1chart:Axis></c1chart:Axis>标记中间，设置AnnoFormat以改变沿着X-轴显示的值从数值格式为货币格式，设置AnnoAngle属性以旋转X-轴标注为逆时针旋转60度。	
XAML		

```
<c1chart:Axis AnnoFormat="c" AnnoAngle="60" />
```

在ChartView.AxisY对象的<c1chart:Axis></c1chart:Axis> 标记中间设置Reversed属性以反向显示Y-轴的值。| |
在Public MainPage（）类中添加以下代码以格式化图表坐标轴：Visual Basic' 设置坐标轴标题 C1Chart1.View.AxisY.Title = New TextBlock(New Run("Kitchen Electronics")) C1Chart1.View.AxisX.Title = New TextBlock(New Run("Price")) ' 设置坐标轴范围 C1Chart1.View.AxisX.Min = 0 C1Chart1.View.AxisX.Max = 500 c1Chart1.View.AxisX.MajorUnit = 20 ' 金融货币格式 C1Chart1.View.AxisX.AnnoFormat = "c" ' 坐标轴标注旋转 C1Chart1.View.AxisX.AnnoAngle = "60" C#// 设置坐标轴标题 c1Chart1.View.AxisY.Title= new TextBlock() { Text = "Kitchen Electronics" }; c1Chart1.View.AxisX.Title = new TextBlock() { Text = "Price" }; // 设置坐标轴范围 c1Chart1.View.AxisX.Min = 0; c1Chart1.View.AxisX.Max = 500;
c1Chart1.View.AxisX.MajorUnit = 20;

	// 金融货币格式
	c1Chart1.View.AxisX.AnnoFormat = "c";
	// 坐标轴标注旋转
	c1Chart1.View.AxisX.AnnoAngle=60;	

下一步，第四步：调整图表的外观，您将学习如何通过编程方式自定义图表的外观。

第四步：调整图表的外观

在此最后一步中，您将通过Theme属性调整图表的外观。

为了通过为了通过XAML在在Visual Studio中设置图表的主题：中设置图表的主题：

为了明确地在您的图表中定义 Office2007Blue主题，添加以下的Theme标记至<c1chart:C1Chart>标签，之后代码将如下所示：

XAML

```
<c1chart:C1Chart Margin="0,0,8,8" MinHeight="160" MinWidth="240" Content="C1Chart"

ChartType="Bar" Theme="Office2007Blue">
```

为了通过代码在 Visual Studio 中设置图表的主题：

WPF

Visual Basic

```
C1Chart1.Theme = TryCast(C1Chart1.TryFindResource(New
ComponentResourceKey(GetType(C1Chart), "Office2007Blue")), ResourceDictionary)
```

C#

```
C1Chart1.Theme = c1Chart1.TryFindResource( new
ComponentResourceKey(typeof(C1.WPF.C1Chart.C1Chart), "Office2007Blue")) as

ResourceDictionary;
```

Visual Basicc1Chart1.Theme = ChartTheme.

C#

```
c1Chart1.Theme = ChartTheme.Office2007Blue;
```

至此，您完成了：至此，您完成了：

Office 2007 Blue 主题应用到了 C1Chart 控件。

恭喜完成！您已经完成了WPF版Chart快速入门，在这里您创建了一个图表应用程序，向图表添加了数据，设置了坐标轴的范围，格式化显示了坐标轴的标注，并自定义了图表的外观。