添加趋势线至C1Chart

添加趋势线和添加一个数据系列非常相似。C1Chart将自动基于数值计算并绘制趋势。默认情况下，趋势线将有一个多项式FitType，其Order属性的值为2。下面的代码演示如何添加趋势线：

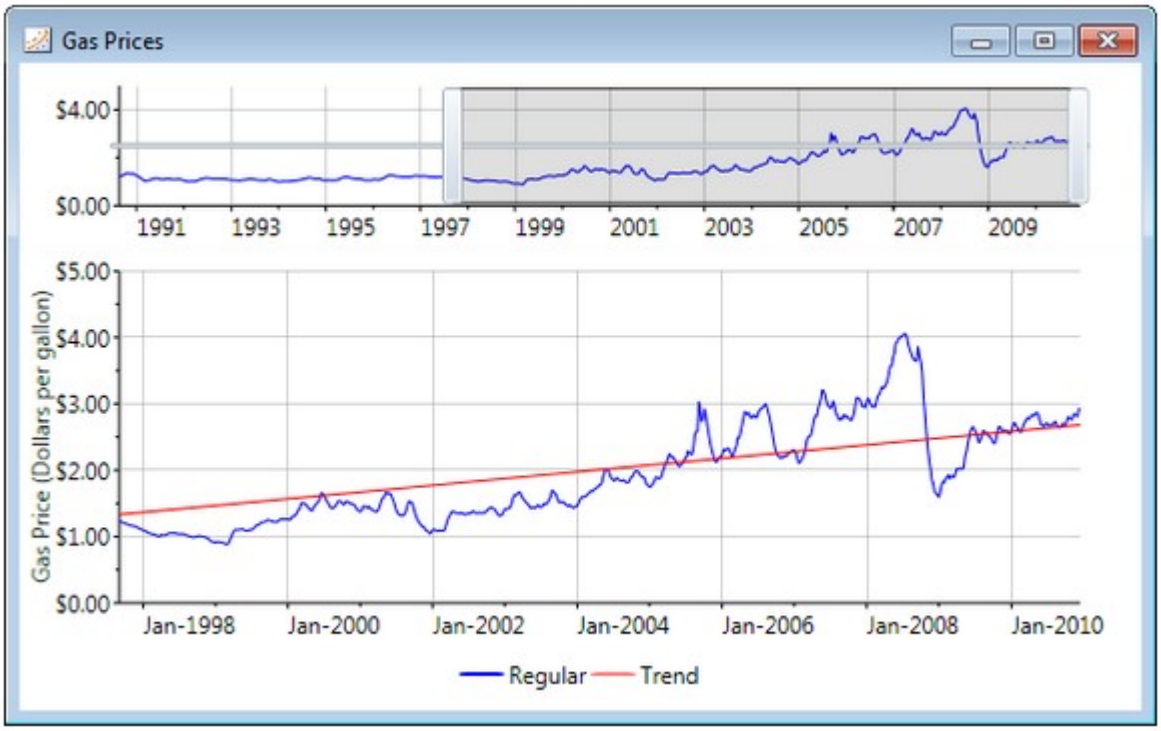
```
Visual Basic
```

```
'add trend line Dim tl As New TrendLine() tl.Label = "Trendline" tl.ConnectionStroke = New SolidColorBrush(Colors.Red)
tl.XValuesSource = myXValues tl.ValuesSource = myValues chart.Data.Children.Add(tl)
```

```
C#
```

```
//添加趋势线
TrendLine tl = new TrendLine(); tl.Label = "Trendline"; tl.ConnectionStroke = new SolidColorBrush(Colors.Red);
tl.XValuesSource = myXValues; tl.ValuesSource = myValues; chart.Data.Children.Add(tl);
```

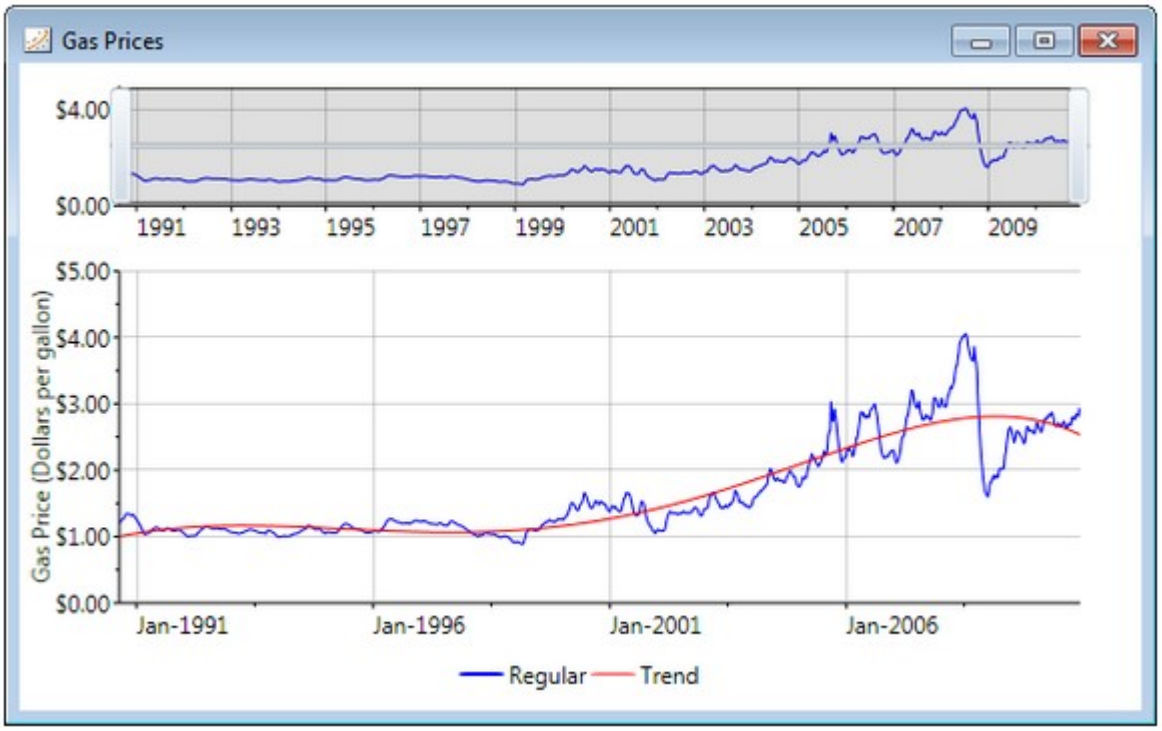
你可以看到下面的图像，默认的趋势线没有紧密地靠近数据：



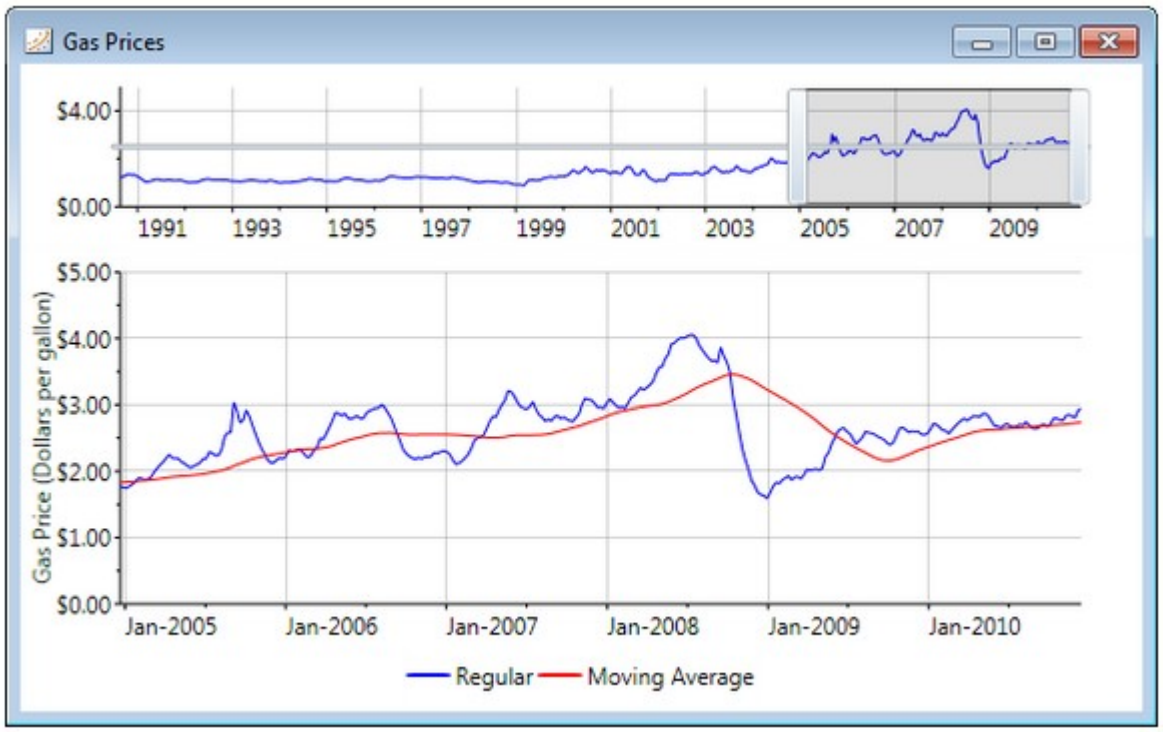
你可以通过改变Order属性来获得更好的拟合。您可以指定FitType以及Order属性以获取您所期望的拟合结果：

```
C#
```

```
TrendLine tl = new TrendLine(); tl.Label = "Trend"; tl.FitType = FitType.Polynom; tl.Order = 6;
```



图中的图表使用的数据展示了汽油的价格，这些值每七天获取一次。对于这一类数据，使用MovingAverage趋势线是比较合适的。在下图中你可以看到它是如何拟合数据的：



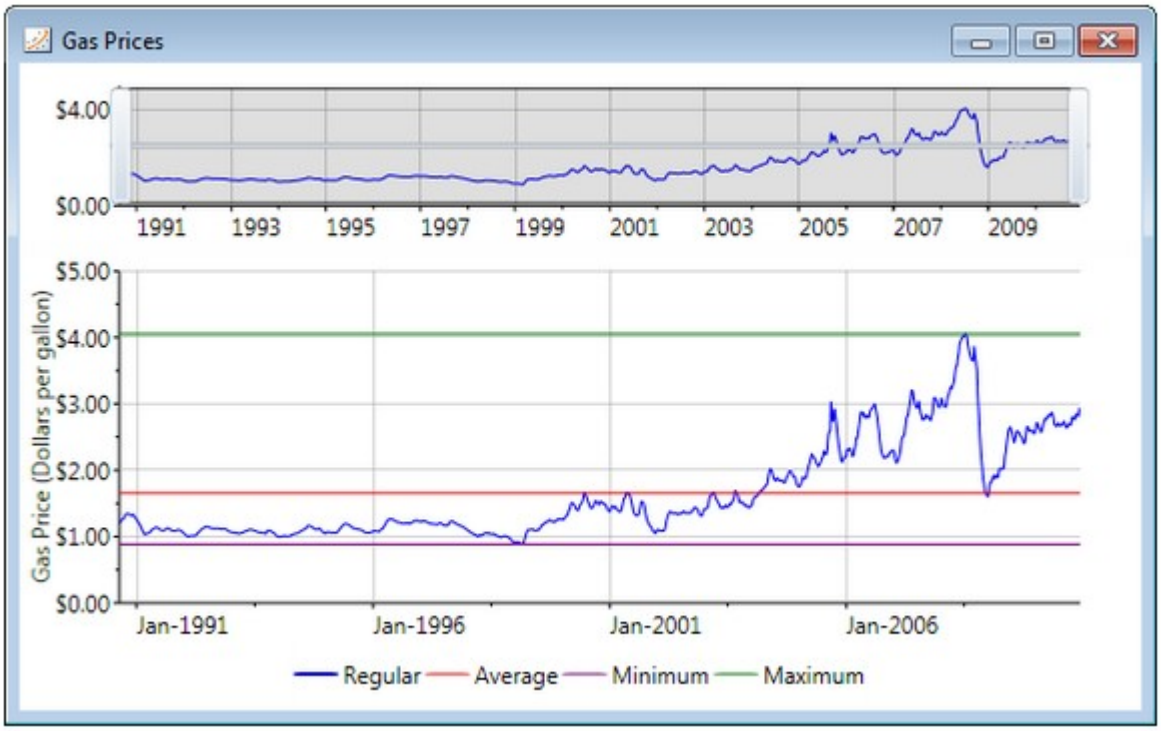
当建立一个移动平均趋势线，你必须实例化一个新的MovingAverage对象并设置Period属性。此属性指定用于趋势线的数据点的个数。由于天然气价格每七天计算一次平均值，每年48个平均值：

C#

```
MovingAverage ma = new MovingAverage(); ma.Label = "Moving Average"; ma.Period = 48; ma.XValuesSource = days;
ma.ValuesSource = price; ma.ConnectionStroke = new SolidColorBrush(Colors.Red); chart.Data.Children.Add(ma);
```

非回归趋势线

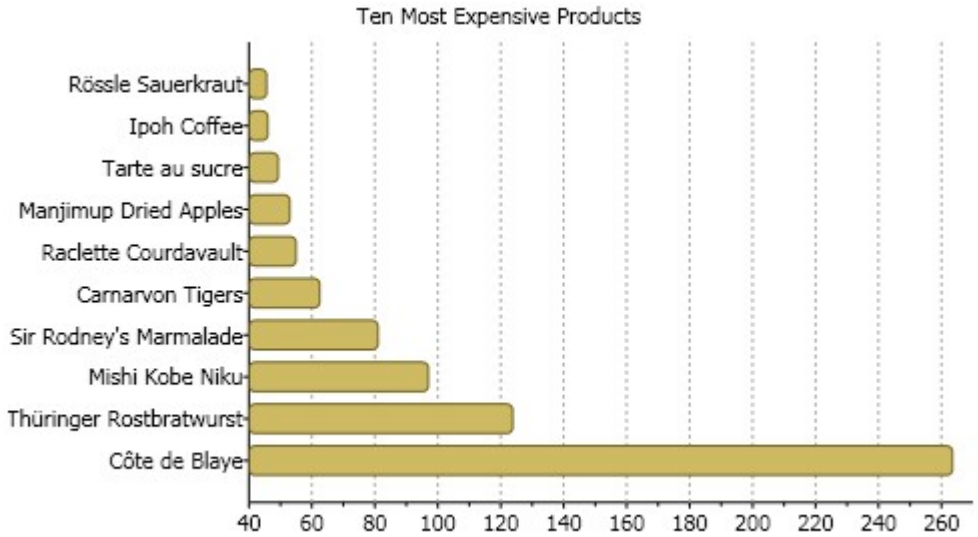
如果您希望高亮显示您简单数据，比如说图表数据的最小值，最大值以及平均值，您可以实例化三个趋势线并设置其FitType属性的值为MinmunX，MaximumX以及AverageX：



教程
以下部分介绍如何为图表绑定数据。
数据绑定教程
以下部分包括C1Chart的数据绑定教程。本教程提供一步一步的说明。按照本章中所概述的步骤，你将能够把C1Chart绑定到数据表和XML文件。
以下两个教程的核心属性如下：
ItemsSource（在线文档 'ItemsSource 属性'）- 提供一个对象列表
ItemNameBinding（在线文档 'ItemNameBinding 属性'）-指定一个元素的名称，比如说X轴上的标签。
ValueBinding（在线文档 'ValueBinding 属性'）- 指定数值型值 声明式绑定到数据表

注意：注意：本教程适用于WPF应用程序

本教程提供了一步一步的指示绑定C1Chart
到数据集的声明。数据将信息作为简单的条形图显示，有一个y-轴表示产品的名称以及一个x-轴表示每一个产品的单价。产品最贵的十种产品的单价以降序方式显示。条形图通过一个系列绘制单价。没有使用图例，因为仅有一个系列。
图表使用来自于Access的示例数据库，Nwind.mdb。本教程假定数据库文件Nwind.mdb位于
Documents\ComponentOne Samples\Common目录，并且为了简洁起见，通过其文件名，而不是其完整路径名引用。
完成本教程将产生一个图表，看起来像以下：



为绑定到C1Chart到数据表声明，完成以下步骤：

1. 在Visual Studio中创建一个新的WPF项目。有关创建WPF项目的更多信息，参见WPF入门。
2. 添加到C1.WPF.C1Chart引用至您的项目。
3. 添加C1Chart控件至窗体。更多信息请参见ComponentOne Studio WPF 版本入门。
4. 一旦C1Chart控制放在窗体上，则将自动生成下面的XAML代码：

```
XAML
Title="Window1" Height="300" Width="500" xmlns:c1chart="clr-
```

```
namespace:C1.WPF.C1Chart;assembly=C1.WPF.C1Chart" Loaded="Window_Loaded"> <Grid>
<c1chart:C1Chart Content="" Margin="10,10,10,18" Name="c1Chart1">
<c1chart:C1Chart.Data>
<c1chart:ChartData>
<c1chart:ChartData.ItemNames>P1 P2 P3 P4
P5</c1chart:ChartData.ItemNames>
<c1chart:DataSeries Label="Series 1" Values="20 22 19 24 25" />
<c1chart:DataSeries Label="Series 2" Values="8 12 10 12 15" />
</c1chart:ChartData>
</c1chart:C1Chart.Data>
<c1chart:Legend DockPanel.Dock="Right" />
</c1chart:C1Chart>
</Grid>
```

5. 选择XAML选项卡，添加下面的XAML代码的命名空间：

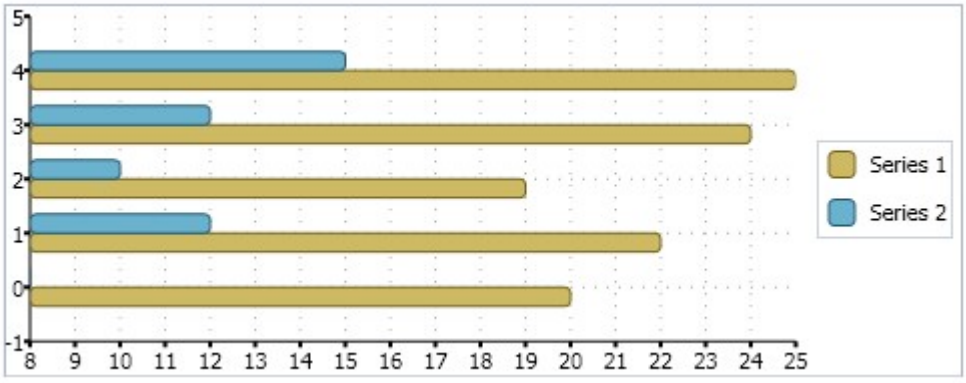
```
XAML
xmlns:sys="clr-namespace:System;assembly=mscorlib"
```

6. 在XAML标记中，更改标题的宽度从300到500。
7. 在<c1chart:C1Chart> 标签内部修改Margin为"0" 并设置ChartType 为 "Bar"。这将改变默认图表的外观从柱形到条形。您的XAML标记应该如下所示：

```
XAML
<Grid>
<c1chart:C1Chart Margin="0" Name="c1Chart1" ChartType="Bar" Height="185" VerticalAlignment="Top">

<c1chart:C1Chart.Data>
<c1chart:ChartData>
<c1chart:DataSeries Label="Series 1" Values="20 22 19 24 25" />
<c1chart:DataSeries Label="Series 2" Values="8 12 10 12 15" />
</c1chart:ChartData>
</c1chart:C1Chart.Data>
<c1chart:Legend DockPanel.Dock="Right" />
</c1chart:C1Chart>
<TextBlock DockPanel.Dock="Top" Text="Ten Most Expensive Products"
HorizontalAlignment="Center"/>
</Grid>
```

您的图表将显示如下：



8. 在c1chart:C1Chart标签结束位置创建一个标签，并将其标签内容设置为"最贵的十种产品"。

```
XAML
<TextBlock DockPanel.Dock="Top" Text="Ten Most Expensive Products"

HorizontalAlignment="Center"/>
```

您的XAML标记现在应该看起来类似以下代码所示：

```
XAML
```

```
<Grid>
<c1chart:C1Chart Margin="0" Name="c1Chart1" ChartType="Bar" Height="185" VerticalAlignment="Top">

<c1chart:C1Chart.Data>
<c1chart:ChartData>
<c1chart:DataSeries Label="Series 1" Values="20 22 19 24 25" />
<c1chart:DataSeries Label="Series 2" Values="8 12 10 12 15" />
</c1chart:ChartData>
</c1chart:C1Chart.Data>
<c1chart:Legend DockPanel.Dock="Right" />
</c1chart:C1Chart>
<TextBlock DockPanel.Dock="Top" Text="Ten Most Expensive Products"
HorizontalAlignment="Center"/>
</Grid>
```

9. 在后台代码文件添加以下using/imports代码：

Visual Basic

```
//WPF
Imports System.Data
Imports System.Data.OleDb

Imports C1.WPF.C1Chart
```

C#

```
//WPF
using System.Data; using System.Data.OleDb; using C1.WPF.C1Chart;
```

10. 在Window_Loaded 函数以外为DataSet声明变量，接下来添加以下代码以获取来自于数据库的产品和单价信息：

Visual Basic

```
Private _dataSet As DataSet
Private Sub Window_Loaded(ByVal sender As Object, ByVal e As RoutedEventArgs)

' 创建连接并填充数据集
Dim mdbFile As String = "c:\Program Files\ComponentOne Studio.NET
2.0\Common\nwind.mdb"
Dim connString As String = String.Format("Provider=Microsoft.Jet.OLEDB.4.0;
Data Source={0}", mdbFile)
Dim conn As New OleDbConnection(connString)
Dim adapter As New OleDbDataAdapter("SELECT TOP 10 ProductName, UnitPrice" &
Chr(13) & "" & Chr(10) & " FROM Products ORDER BY UnitPrice DESC;", conn)
_dataSet = New DataSet()
adapter.Fill(_dataSet, "Products")
' 设置图表数据的源
c1Chart1.Data.ItemsSource = _dataSet.Tables("Products").Rows
End Sub
```

C#

```
DataSet _dataSet; private void Window_Loaded(object sender, RoutedEventArgs e)
{
// 创建连接并填充数据集
string mdbFile = @"c:\Program Files\ComponentOne Studio.NET
2.0\Common\nwind.mdb";
string connString = string.Format("Provider=Microsoft.Jet.OLEDB.4.0; Data
Source={0}", mdbFile);
OleDbConnection conn = new OleDbConnection(connString); OleDbDataAdapter
adapter =
new OleDbDataAdapter(@"SELECT TOP 10 ProductName, UnitPrice
FROM Products ORDER BY UnitPrice DESC;", conn);
_dataSet = new DataSet(); adapter.Fill(_dataSet, "Products");
// 设置图表数据的源
<ac:structured-macro ac:name="unmigrated-wiki-markup" ac:schema-version="1"
ac:macro-id="95ce8aa4-7a11-4e78-993d-8f24b0d99250"><ac:plain-text-body><![CDATA[
c1Chart1.Data.ItemsSource = _dataSet.Tables["Products"].Rows; }
```

```
]]></ac:plain-text-body></ac:struc
```

注意注意：请确保mdbFile文件的路径匹配nwind.mdb数据库文件在您本机的实际位置。

11. 在XAML选项卡以切换至XAML视图，接下来删除以下来自于ChartData的默认数据：

```
XAML
<c1chart:ChartData.ItemNames>P1 P2 P3 P4 P5</c1chart:ChartData.ItemNames>
```

<c1chart:DataSeries Label="Series 1" Values="20 22 19 24 25" />
<c1chart:DataSeries Label="Series 2" Values="8 12 10 12 15" />
现在C1Chart控件在窗体上显示为空白。

12. 在<c1chart:C1Chart.Data>标签内部添加ItemNameBinding
至ChartData以指定元素的名称，也就是显示在y-轴上的标签，同时设置ValueBinding属性至DataSeries，以便为系列指定数值类型的值。

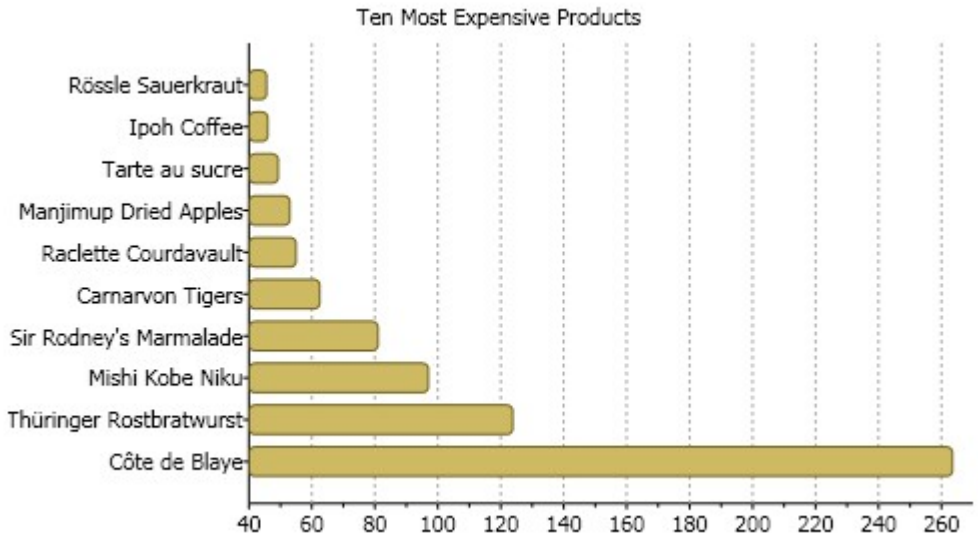
```
XAML
<c1chart:ChartData ItemNameBinding=
"{Binding Path=[ProductName]}"
>
<c1chart:DataSeries ValueBinding=
"{Binding Path=[UnitPrice]}"
/>
</c1chart:ChartData>
```

您工程中的XAML代码应当看起来如以下所示：

```
XAML
<Window x:Class="Chart for WPF_QuickStart.Window1" xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation
" xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" xmlns:sys="clr-namespace:System;assembly=mscorlib"
Title="Window1" Height="300" Width="500" Loaded="Window_Loaded" xmlns:c1chart="clr-namespace:C1.WPF.C1Chart;assembl
y=C1.WPF.C1Chart"> <Grid>
<c1chart:C1Chart Margin="0" Name="c1Chart1" ChartType="Bar">
<TextBlock DockPanel.Dock="Top" Text="Ten Most Expensive Products"
HorizontalAlignment="Center"/>
<c1chart:C1Chart.Data>
<c1chart:ChartData ItemNameBinding="{Binding Path=
[ProductName]}"
>
<c1chart:DataSeries ValueBinding="{Binding Path=
[UnitPrice]}"
/>
</c1chart:ChartData>
</c1chart:C1Chart.Data>
</c1chart:C1Chart>
</Grid>
</Window>
```

13. 从XAML中删除<c1chart:Legend DockPanel.Dock="Right" />标签，以便删除内置的图例控件。
14. 运行您的工程，以确保一切正常工作。

在运行时可以看到以下效果在运行时可以看到以下效果图表显示来自产品表的数据。



绑定到一个XML

本教程提供了一个将XML作为数据到嵌入到XAML页面以绑定ClChart控件至该XML数据的分步说明。该数据将信息显示为一个简单的条形图，y-轴表示城市的名称，x-轴表示每一个城市的人口。该条形图使用一个系列绘制人口信息。图例区用于显示人口的颜色。

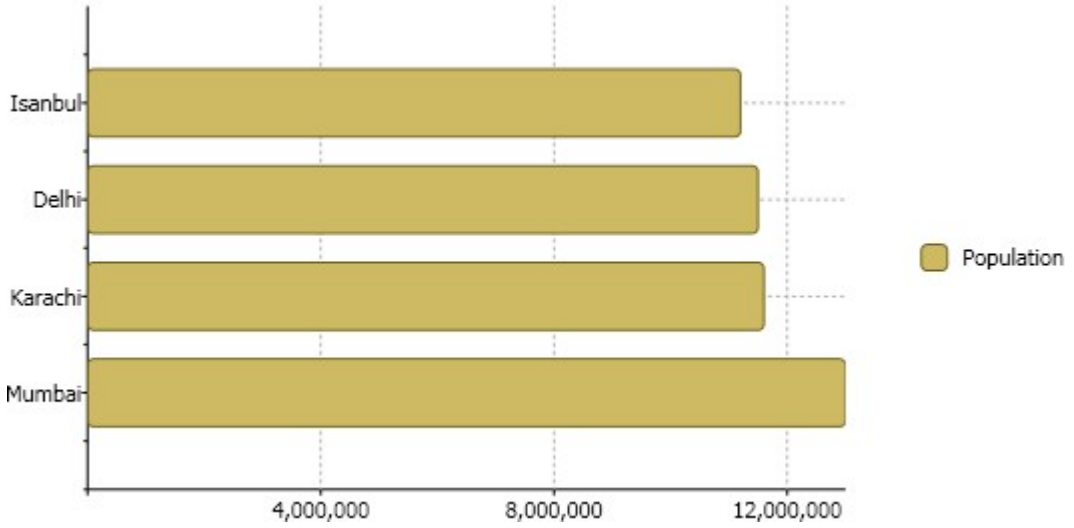
本教程中的绑定设置在ChartData类上，通过以下XAML代码：

XAML

```
<clchart:ChartData ItemsSource="{Binding Source={StaticResource data}}"

ItemNameBinding="{Binding XPath=CityName}">
<clchart:DataSeries Label="Population"
ValueBinding="{Binding XPath=Population}" />
</clchart:ChartData>
```

完成本教程将产生一个图表，看起来像以下：



绑定绑定 ClChart 至至XML：：

1. a. 在Visual Studio中创建一个新的WPF工程。有关创建WPF项目的更多信息，参见WPF入门。
   b. 在资源部分中，将数据直接嵌入到数据岛。XML数据岛必须包含在在<x:Xdata>标签中，并且总是有一个根节点，在本示例中，该根节点为Cities：

   XAML

```
<Grid.Resources>
<XmlDataProvider x:Key="data" XPath="Cities/City">

<x:XData>
<Cities xmlns="">
<City>
<CityName>Mumbai</CityName>
<Population>13000000</Population>
</City>
<City>
<CityName>Karachi</CityName>
<Population>11600000</Population>
</City>
<City>
<CityName>Delhi</CityName>
<Population>11500000</Population>
</City>
<City>
<CityName>Istanbul</CityName>
<Population>11200000</Population>
</City>
</Cities>
</x:XData>
</XmlDataProvider>
</Grid.Resources>
```

c. 向您的工程添加到 C1.WPF.C1Chart 的引用。
d. 添加 C1Chart控件窗体。

一旦C1Chart控件添加到窗体，将会生成以下XAML代码：

XAML

```
Title="Window1" Height="50" Width="100" xmlns:c1chart="clrnamespace:C1.WPF.C1Chart;assembly=C1.WPF.C1Chart" Loaded="Window
_Loaded"> <Grid>
<c1chart:C1Chart Content="" Margin="10,10,10,18" Name="c1Chart1">
<c1chart:C1Chart.Data>
<c1chart:ChartData>
<c1chart:ChartData.ItemNames>P1 P2 P3 P4
P5</c1chart:ChartData.ItemNames>
<c1chart:DataSeries Label="Series 1" Values="20 22 19 24 25" />
<c1chart:DataSeries Label="Series 2" Values="8 12 10 12 15" />
</c1chart:ChartData>
</c1chart:C1Chart.Data>
<c1chart:Legend DockPanel.Dock="Right" />
</c1chart:C1Chart>
</Grid>
```

1. a. 改变窗体的宽度为 "300" ，高度为 "550"
   b. 在<c1chart:C1Chart> 标签内部修改Margin为"0" 并设置ChartType 为 "Bar"。这将改变默认图表的外观从柱形到条形。你的XAML代码应该看起来如以下所示：

XAML

```
<c1chart:C1Chart Margin="0" Name="c1Chart1" ChartType="Bar">


</c1chart:C1Chart>
```

c. 在XAML文件中找到<c1chart:C1Chart.Data>标签，并从中删除以下XAML代码：

XAML

```
<c1chart:ChartData.ItemNames>P1 P2 P3 P4 P5</c1chart:ChartData.ItemNames>

<c1chart:DataSeries Label="Series 1" Values="20 22 19 24 25" />
<c1chart:DataSeries Label="Series 2" Values="8 12 10 12 15" />
```

两个默认的系列从C1Chart移除，现在C1Chart显示为空白，因为它没有任何数据。

d. 在<c1chart:C1Chart.Data>标签内部，向ChartData 添加ItemNameBinding属性以指定元素的名称，在本示例中，这是显示在y-轴上的标签，同时指定DataSeries的ValueBinding 属性以指定系列的数值型的值。以下实例通过绑定扩展绑定ChartData.ItemsSource属性，指定数据源。ChartData.ItemName Binding属性通过使用绑定扩展指定Path进行绑定。DataSeries.Label属性通过使用绑定扩展进行绑定，指定Path，在这里是 Population。

```
XAML
<c1chart:ChartData ItemsSource="{Binding Source={StaticResource data}}"
ItemNameBinding="{Binding XPath=CityName}"> <c1chart:DataSeries Label="Population" ValueBinding="{Binding

XPath=Population}" />
</c1chart:ChartData>
```

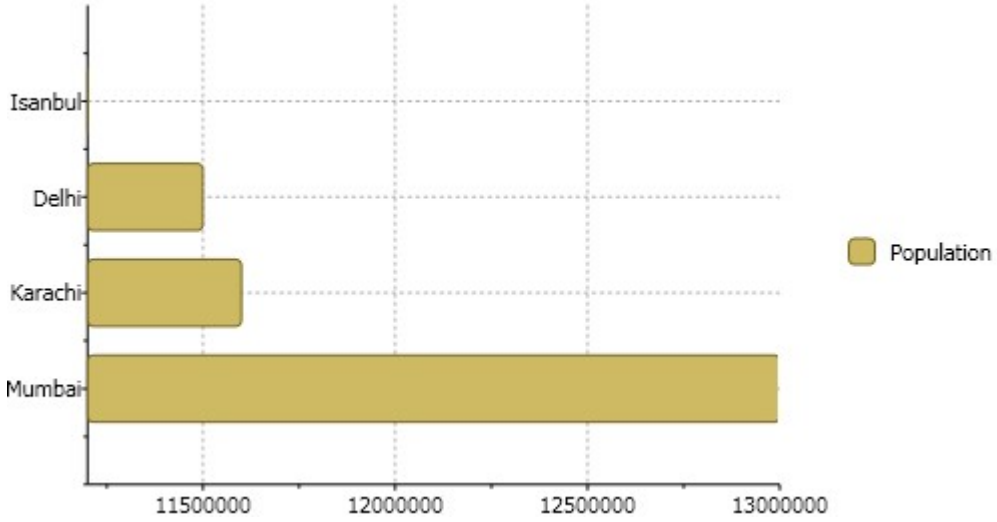
您的C1Chart的XAML代码应当看起来如下所示：

```
XAML
<c1chart:C1Chart Height="300" HorizontalAlignment="Left" Margin="0"
Name="c1Chart1" ChartType="Bar" VerticalAlignment="Top" Width="500">
<c1chart:C1Chart.Data>
<c1chart:ChartData ItemsSource="{Binding Source={StaticResource data}}"

ItemNameBinding="{Binding XPath=CityName}">
<c1chart:DataSeries Label="Population"
ValueBinding="{Binding XPath=Population}" />
</c1chart:ChartData>
</c1chart:C1Chart.Data>
<c1chart:Legend DockPanel.Dock="Right" />
</c1chart:C1Chart>
```

e. 运行您的工程，以确保一切正常工作。

您的图表将看起来如下所示：



请注意这里x-轴的标注是如何显示的。我们需要格式化x-轴的标注使得人口的数值显示千位分隔符。

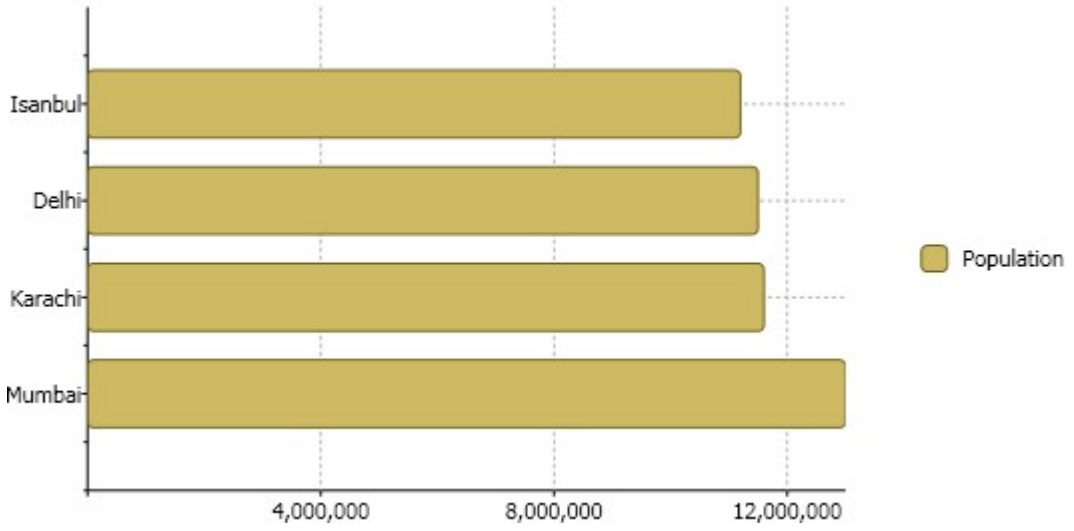
1. a. 为C1Chart的ChartView.AxisX属性声明标签。您将需要设置AxisX 的以下属性的以格式化标注以及网格线。

在结束标签</c1chart:C1Chart.Data>之后添加以下XAML代码：

```
XAML
<c1chart:C1Chart.View>
<c1chart:ChartView>
<c1chart:ChartView.AxisX >
<c1chart:Axis Min="0" MajorGridStroke="DarkGray"

AnnoFormat="#,###,###"/>
</c1chart:ChartView.AxisX>
</c1chart:ChartView>
</c1chart:C1Chart.View>
```

图表上的x-轴的标注将更新显示为以下：


Isanbul
Delhi
Karachi
Mumbai
4,000,000
8,000,000
12,000,000
Population