使用MVVM

WPF及Silverlight版Chart支持MVVM（模型-视图-ViewModel）设计模式。整个图表可以被声明式地通过源生的绑定技术绑定到XAML。下面的步骤演示了如何在一个MVVM架构的WPF或Silverlight应用程序中使用C1Chart（在线文档 'C1Chart 类'）。

1. 创建一个新类命名为Sale，实现INotifyPropertyChanged接口。

C#

```
public class Sale : INotifyPropertyChanged
{ private string _product; private double _value; private double _discount; public Sale(string product, double valu
e, double discount) {
Product = product;
Value = value;
Discount = discount; } public string Product
{ get { return _product; } set { if (_product != value)
{
_product = value;
OnPropertyChanged("Product");
}
} } public double Value
```

```
{ get { return _value; } set { if (_value != value)
{
_value = value;
OnPropertyChanged("Value");
}
} } public double Discount
{ get { return _discount; } set { if (_discount != value)
{
_discount = value;
OnPropertyChanged("Discount");
}
} } public event PropertyChangedEventHandler PropertyChanged; void OnPropertyChanged(string propertyName)
{
if (PropertyChanged != null)
PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
}
}
```

2. 创建一个新的类并命名为SaleViewModel。这将做为包含C1Chart的View的数据上下文。

C#

```
public class SaleViewModel : INotifyPropertyChanged
{ private ObservableCollection<Sale> _sales = new ObservableCollection<Sale>
(); public SaleViewModel()
{
//加载数据
LoadData(); } public ObservableCollection<Sale> Sales
{ get { return _sales; }
} public void LoadData()
{
//TODO: 从数据源加载数据
```

```
_sales.Add(new Sale("Bikes", 23812.89, 12479.44));
_sales.Add(new Sale("Shirts", 79752.21, 19856.86));
_sales.Add(new Sale("Helmets", 63792.05, 16402.94));
_sales.Add(new Sale("Pads", 30027.79, 10495.43));
} public event PropertyChangedEventHandler PropertyChanged; void OnPropertyChanged(string propertyName)

{
if (PropertyChanged != null)
PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
}
}
```

该类将包含一个ObservableCollection，Sales，以及生成模拟数据的初始化方法。

3. 切换到源代码视图并创建一个新的WPF或者Silverlight的用户控件，并命名为 SaleView.xaml。在LayoutRoot Grid之前添加以下XAML：

XAML

```
<UserControl.Resources>
<local:SaleViewModel x:Key="viewModel" />
</UserControl.Resources>
<UserControl.DataContext>
<Binding Source="{StaticResource viewModel}"/>

</UserControl.DataContext>
```

该XAML声明了一个SalesViewModel做为资源，并设置其为UserControl的DataContext。在运行时，View将被绑定到ViewModel。在View中的控件可以绑定到ViewModel的公共属性。

4. 向页面添加一个C1Chart控件。
5. 替换C1Chart默认的XAML为以下代码：

XAML

```
<c1:C1Chart ChartType="Column" Name="c1Chart1" Palette="Module">
<c1:C1Chart.Data>
<c1:ChartData ItemsSource="{Binding Sales}" ItemNameBinding="{Binding Product}">

<c1:DataSeries Label="Value" ValueBinding="{Binding Value}" />
<c1:DataSeries Label="Discount" ValueBinding="{Binding Discount}" />
</c1:ChartData>
</c1:C1Chart.Data>
<c1:C1ChartLegend />
</c1:C1Chart>
```

该XAML定义了一个具有两个数据系列的C1Chart。该ChartData的ItemsSource设置为一个Sales对象的集合，该集合暴露在ViewModel上。每个DataSeries 都有其ValueBinding 属性设置并设置ItemNameBinding 以沿x轴展示我们的产品名称。

注意注意：如果你使用的是XYDataSeries，那么你应该指定每个系列的XValueBinding，而不应该设置

ItemNameBinding。

6. 打开App.xaml.cs应用程序配置文件并将Application_Startup 事件，替换为以下代码：

Example Title

```
private void Application_Startup(object sender, StartupEventArgs e)

{ this.RootVisual = new Views.SaleView();
}
```

该代码设置RootVisual以便在启动时显示SalesView。

7. 运行应用程序并观察到C1Chart 已经绑定到ViewMode上的Sales数据。