Windows Phone Mango编程实践

***Windows Phone Mango Programming Practice***

第二篇 Silverlight交互篇

# 数据绑定

## 概述

Windows Phone应用程序中的控件显示的数据通常是一个业务对象或者对象集合，如股票行情或图像的集合，有时还需要数据在控件中转换。本节参考APP HUB的文章Data Binding to Controls重点讲述如何数据绑定到控件。

Windows Phone的Silverlight数据绑定必须为每个绑定都指定绑定源和绑定目标。下图演示绑定的基本概念。

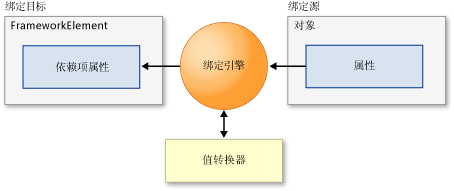


图9-1 基本概念

绑定引擎从绑定源对象获取以下方面的信息：

* 目标 UI 属性，用于显示数据，并且可能允许用户对数据进行更改。
* 包含在源和目标之间流动的数据的源对象。源可以是任何 CLR 对象，包括目标元素本身或者其他 UI 元素。如果目标处于某一数据模板中，则源对象可以是该模板应用的 UI 元素。
* 数据转换即值转换器，属于可选选项。
* 数据流的方向。

每个绑定都有一个Mode 属性，该属性决定数据流动的方式和时间。支持以下三种类型的绑定：

OneTime--创建时绑定，该绑定使用绑定源的数据更新目标。

OneWay--创建时绑定以及每当绑定源的数据发生变化时，该绑定使用绑定源的数据更新绑定目标。这是默认模式。

TwoWay--当绑定目标和绑定源有一个发生变化时，绑定既更新目标也更新源，或者可以设定对源进行更新的时机。

## 绑定单项数据

在典型的数据绑定，从另一个对象的属性自动更新一个对象的属性。

本例中提供数据的对象是滑块控件，滑块控件的Value属性是绑定源的数据，接收的数据的TextBlock的Text属性，以及Rectangle的Width属性是绑定目标。

Silverlight Project: SliderBindings file: MainPage.xaml

<Slider Name="slider"

Value="90"

Grid.Row="0"

Maximum="180"

Margin="24" />

<TextBlock Name="txtblk"

Text="{Binding ElementName=slider, Path=Value}"

Grid.Row="1"

FontSize="48"

HorizontalAlignment="Center"

VerticalAlignment="Center" />

<Rectangle Grid.Row="2"

Width="{Binding ElementName=slider, Path=Value}"

RenderTransformOrigin="0.5 0.5"

Fill="Blue">

<Rectangle.RenderTransform>

<RotateTransform x:Name="rotate"

Angle="90" />

</Rectangle.RenderTransform>

</Rectangle>

运行结果如图9-2 Slider绑定，拖动滑块文本框显示的数据和矩形的高度都由于数据绑定发生变化。

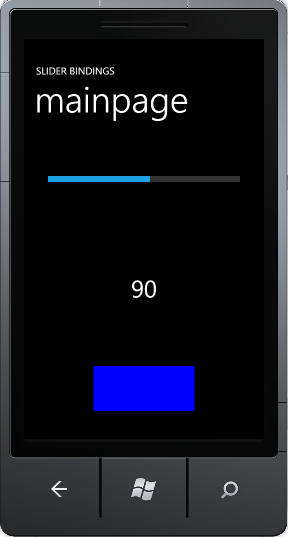
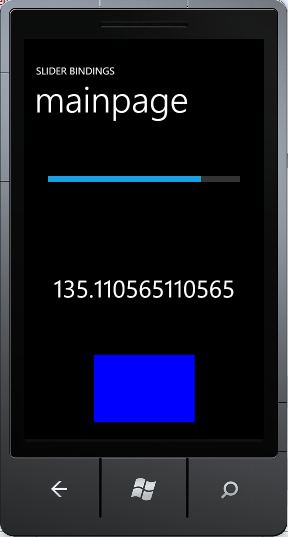
 

图9-2 Slider绑定

## 使用数据模板绑定数据对象集

前面的例子说明将单项数据绑定到控件，A more common scenario is to bind to a collection of business objects.更常见的情况是将对象集数据绑定到控件。参考**APP HUB**上的**Data Binding to Controls**，实现了本节中的实现数据模板绑定数据对象集，并对显示的日期格式做了数据转换。

为了使绑定源对象的更改能够传播到绑定目标，绑定源必须实现 INotifyPropertyChanged 接口。INotifyPropertyChanged 具有 PropertyChanged事件，该事件通知绑定引擎源已更改，以便绑定引擎可以更新目标值。

在实现绑定对象集合之前，应考虑使用 ObservableCollection<T> 类，该类具有 INotifyCollectionChanged 和 INotifyPropertyChanged 的内置实现。The following example binds a collection of music

### 定义数据源

创建定义集合中的每个对象的类。在MainPage.xaml.cs中声明Recording类。

Silverlight Project: ComboBoxBingds file: MainPage.xaml.cs

public class Recording

{

public Recording() { }

public Recording(string artistName, string cdName, DateTime release)

{

Artist = artistName;

Name = cdName;

ReleaseDate = release;

}

public string Artist { get; set; }

public string Name { get; set; }

public DateTime ReleaseDate { get; set; }

// Override the ToString method.

public override string ToString()

{

return Name + " by " + Artist + ", Released: " + ReleaseDate.ToShortDateString();

}

}

在MainPage.xaml.cs中添加引用。

Silverlight Project: ComboBoxBingds file: MainPage.xaml.cs

using System.Collections.ObjectModel;

using System.Windows.Data;

创建对象集合。在MainPage.xaml.cs中添加Recording的数据集。

Silverlight Project: ComboBoxBingds file: MainPage.xaml.cs

public partial class MainPage : PhoneApplicationPage

{

public ObservableCollection<Recording> MyMusic = new ObservableCollection<Recording>();

// Constructor

public MainPage()

{

InitializeComponent();

// Add items to the collection.

MyMusic.Add(new Recording("Chris Sells", "Chris Sells Live",

new DateTime(2008, 2, 5)));

MyMusic.Add(new Recording("Luka Abrus",

"The Road to Redmond", new DateTime(2007, 4, 3)));

MyMusic.Add(new Recording("Jim Hance",

"The Best of Jim Hance", new DateTime(2007, 2, 6)));

// Set the data context

LayoutRoot.DataContext = new CollectionViewSource { Source = MyMusic };

}

}

### 创建数据模板

1. 创建DataTemplate 元素。

**注意：**

可以直接在控件上设置 ItemTemplate，但是如果将其作为资源创建，则模板可重用。

1. 将 ItemsSource 绑定到数据源。
2. 添加详细查看。

Silverlight Project: ComboBoxBingds file: MainPage.xaml

<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">

<ComboBox x:Name="ComboWithTemplate" ItemsSource="{Binding}"

Foreground="Black" FontSize="18" Height="50" Width="400" >

<ComboBox.ItemTemplate>

<DataTemplate>

<StackPanel Orientation="Horizontal" Margin="2">

<TextBlock Text="Artist:" Margin="2" />

<TextBlock Text="{Binding Artist}" Margin="2" />

<TextBlock Text="CD:" Margin="10,2,0,2" />

<TextBlock Text="{Binding Name}" Margin="2" />

</StackPanel>

</DataTemplate>

</ComboBox.ItemTemplate>

</ComboBox>

<!--The UI for the details view-->

<StackPanel x:Name="RecordingDetails">

<TextBlock FontWeight="Bold" Text="{Binding Artist, Mode=OneWay}" Margin="5,0,0,0"/>

<TextBlock FontStyle="Italic" Text="{Binding Name, Mode=OneWay}" Margin="5,0,0,0"/>

<TextBlock Text="{Binding ReleaseDate,

Mode=OneWay,

Converter={StaticResource StringConverter},

ConverterParameter=Released: \{0:d\}}"

Margin="5,0,0,0" />

</StackPanel>

### 数据转换

通过创建一个类和实现 IValueConverter 接口来针对每个具体的应用场景自定义转换器。

Silverlight Project: ComboBoxBingds file: MainPage.xaml.cs

public class StringFormatter : IValueConverter

{

// This converts the value object to the string to display.

// This will work with most simple types.

public object Convert(object value, Type targetType,

object parameter, System.Globalization.CultureInfo culture)

{

// Retrieve the format string and use it to format the value.

string formatString = parameter as string;

if (!string.IsNullOrEmpty(formatString))

{

return string.Format(culture, formatString, value);

}

// If the format string is null or empty, simply

// call ToString() on the value.

return value.ToString();

}

// No need to implement converting back on a one-way binding

public object ConvertBack(object value, Type targetType,

object parameter, System.Globalization.CultureInfo culture)

{

throw new NotImplementedException();

}

}

如果为绑定定义了 Converter 参数，则绑定引擎会调用 Convert 和 ConvertBack 方法。从源传递数据时，绑定引擎调用 Convert 并将返回的数据传递给目标。从目标传递数据时，绑定引擎调用 ConvertBack 并将返回的数据传递给源。

Silverlight Project: ComboBoxBingds file: MainPage.xaml

<phone:PhoneApplicationPage

……

xmlns:local="clr-namespace:ComboBoxBingds"

……

>

<phone:PhoneApplicationPage.Resources>

<local:StringFormatter x:Key="StringConverter"/>

</phone:PhoneApplicationPage.Resources>

……

<TextBlock Text="{Binding ReleaseDate,

Mode=OneWay,

Converter={StaticResource StringConverter},

ConverterParameter=Released: \{0:d\}}"

Margin="5,0,0,0" />

……

运行结果如图9-4 数据绑定运行结果。

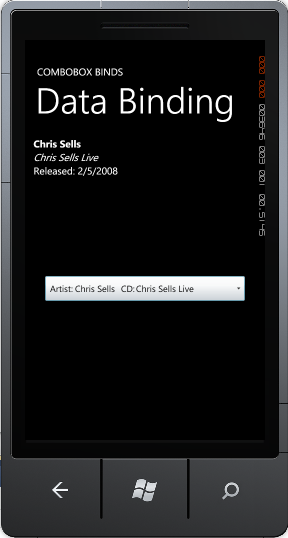
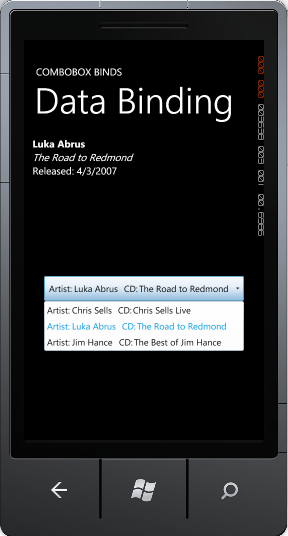
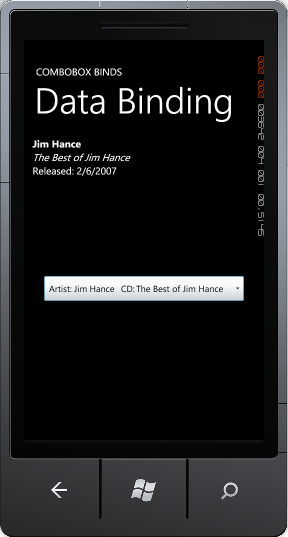
  

图9-4 数据绑定运行结果