Windows 8 开发 31 日

第 05 日

设置合约

译者：BeyondVincent(破船)

时间：2013.4.23

版本： 2.0

关于破船

程序猿砌墙于云南昆明!

长期扎根移动软件开发!

爱跑步爱打篮球爱运动!

命中无大富大贵之面相!

愿健康与平淡相随一生!

你可以发邮件与破船取得联系: BeyondVincent@gmail.com

还可以关注破船的微博: 腾讯微博和新浪微博。

这里是破船的个人博客，欢迎光临：破船之家

关于 Windows 8 开发 31 日翻译

Windows 8 开发 31 日是由 Jeff Blankenburg 和 Clark Sell 原创的。

官方站点：http://31daysofwindows8.com/

涉及到两个版本：

XAML/C#（由 Jeff Blankenburg 撰写）

HTML5/JS （由 Clark Sell 撰写）

其中涉及到的资源和相关代码请到这里下载：

https://github.com/csell5/31DaysOfWindows8

在这里，由于破船对 HTML5/JS 不熟悉，所以只翻译 XAML/C#相关主题。

建议大家前往看原创内容，如果看不明白，再来这里看我翻译的相关内容。

如果翻译不正确的地方，可以通过上面的联系方式告诉破船。

破船祝你阅读愉快！

目录

关于破船 2

关于 Windows 8 开发 31 日翻译 3

目录 4

第 05 日设置合约 5

1.0. 介绍 5
1.1. 开始示例 6
1.2. 创建 Popup 控件 8
1.3. 创建 UserControl 10
1.4. 保存数据 13
1.5. 总结 14

第 05 日设置合约

1.0. 介绍

今天我们开始学习合约，关于合约会有好几篇文章，我将从设置合约开始。在程序中，设置合约的使用频率是蛮高的，也非常重要。下面我们先来谈谈合约：

合约：合约就像一个或多个程序之间的协议。合约定义了应用程序在与其它程序或 Windows 进行交互时必须遵循的一些约定。

例如，Windows 允许用户将内容从一个程序共享到另外一个程序。共享内容的程序通过满足指定的要求来支持源合约，而接收共享内容的程序通过满足另外的一组要求来支持目标合约。两种类型的程序都需要知道另外一个程序的相关信息。参与共享合约的每一个程序，都能满足共享工作流的支持。

这听起来有点像是一个被管理的接口——在程序和操作系统之间。在我们的程序中有 5 个合约可以使用：

- File Picker

- Play To
- Search
- Settings
- Share

就像典型的编程接口一样，即扩展这些接口功能的程序并不知道这些接口的原理。

在 Windows 8 中，所有的这些实际感觉就像是发生在用户体验级别的。我的意思是什么呢？我举共享合约的一个例子来说明一下吧。简单来说，应用程序告诉操作系统并进行注册，比如共享，可以是接收一个图片或者共享一个图片。之后 Windows 将作为中间人，进行相关的处理。

想象一下，有这样的一个场景，当你在用 IE 浏览网页的时候，发现一篇非常好的文章。你想将文章通过 email 或者 Twitter 分享给别人。加入你已经装了 email 或 Twitter 程序，并且 email 或 Twitter 都接受分享一个 URI，那么 IE 就可以通过 email 或 Twitter 进行分享。更好的是 ,email 或 Twitter 程序不一定就需要运行起来，Windows 会帮忙处理其中的事情。

听起来有点像典型的编程接口，不是吗？

1.1. 开始示例

如果你看看本系列以外的许多示例，那么你会发现有一组复杂的方法在许多页面间被使用，如果你像我一样，那么会相当迷惑。而我写的这一系列文章旨在简单的理解和使用那些方法。那么我们开始吧。

在程序启动的时候，App.xaml.cs 文件中有一个名为 OnLaunched 的方法会被

调用。在 OnLaunched 的开头，我创建了一个 event handler，当用户打开 SettingsPane 时，会被调用。你自定义的设置画面直到用户打开 Settings Charm 时，才会被加载。当你“暂停”你的程序或者游戏，如果设置面板被打开了，那么这个 event 也会被调用。

```
protectedoverridevoid OnLaunched(LaunchActivatedEventArgs args)
{
SettingsPane.GetForCurrentView().CommandsRequested += App_CommandsRequested;

Frame rootFrame = Window.Current.Content asFrame;
      ......
```

添加了 event handler 之后，就可以添加 SettingsCommands 到我们的 SettingsPane 上，创建 SettingsCommand 的几个步骤如下：

创建一个新的 SettingsCommand。这需要三个属性：一个 ID、一个 lable 和一个 action，其中当 command 按下时，action 会被执行，在我的示例中，该 action 将创建一个 Popup 控件，然后我会创建一个 UserControl，放置在 Popup 中，最后将其滑入屏幕中显示出来。

处理 Popup 的解雇(dismissal)。当用户使用完了设置 那么我们还需要将 Popup 解雇，并返回到程序中。

如果你还没有添加 event handler 的话，那么你需要一个名为 App_CommandsRequested 的 event handler，当然名称可以随便命名。在这个 handler 中，我将获取到 SettingPane，并添加一个 command，如下代码：

```
void App_CommandsRequested(SettingsPane sender, SettingsPaneCommandsRequestedEventArgs args)
{
```

```
SettingsCommand command = newSettingsCommand("about", "About This App", (handler) =>
      {
Popup popup = BuildSettingsItem(newAboutPage(), 646);
            popup.IsOpen = true;
      });

      args.Request.ApplicationCommands.Add(command);
}
```

如上代码所示，我创建了一个 SettingsCommand 对象，并提供了三个值。除非你需要在程序运行时修改 SettingsPane，否则第一个参数值并不太重要。它就是一个简单 ID，随后可以通过这个 ID 可以引用到 SettingsCommand。标签“About This App”可以是任意的字符串，不过我建议不要超过 40 个字符，否则会被截断。最后一个参数值是这个 command 的 handler。当用户轻触 lable 时，会执行该 handler。在这里，我用 lambada 表达式来简化该处理。表达式里面，我创建了一个 Popup 控件，并将其 IsOpen 属性设置为 true。改变这个属性，可以使 Popup 显示在屏幕上。

1.2. 创建 Popup 控件

在这节里面，来看看 BuildSettingsItem 方法。在上一节中的代码，你可能已经注意到了，我传递了两个值给 BuildSettingsItem 函数。首先是一个自定义的 UserControl，之后我会介绍到该 UserControl，现在只需要明白 AboutPage.xaml 是一个 UserControl 即可，AboutPage.xaml 是用于 Popup 控件中的。

```
privatePopup BuildSettingsItem(UserControl u, int w)
{
Popup p = newPopup();
      p.IsLightDismissEnabled = true;
```

```
        p.ChildTransitions = newTransitionCollection();
        p.ChildTransitions.Add(newPaneThemeTransition()
        {
            Edge = (SettingsPane.Edge == SettingsEdgeLocation.Right) ?
EdgeTransitionLocation.Right :
EdgeTransitionLocation.Left
        });

        u.Width = w;
        u.Height = Window.Current.Bounds.Height;
        p.Child = u;

        p.SetValue(Canvas.LeftProperty, SettingsPane.Edge == SettingsEdgeLocation.Right ?
(Window.Current.Bounds.Width - w) : 0);
        p.SetValue(Canvas.TopProperty, 0);

return p;
}
```

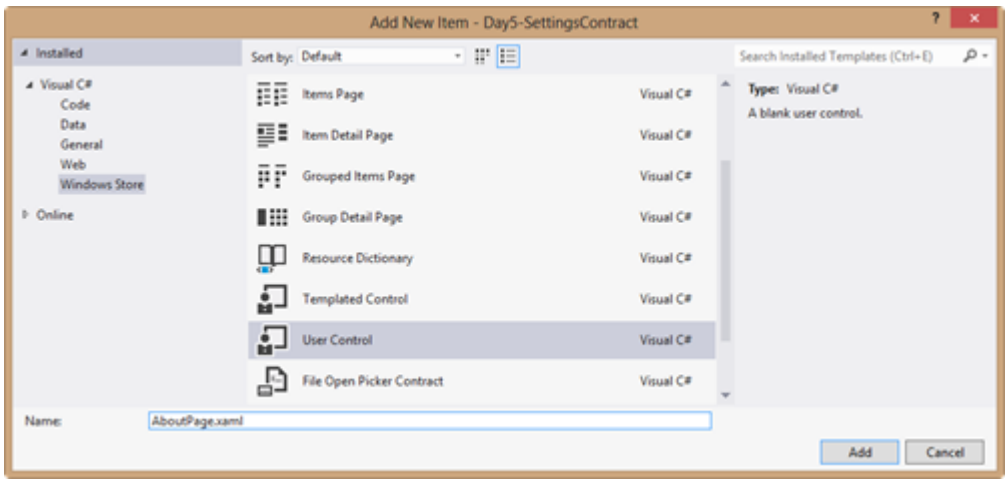
上面的代码看着比较凌乱，它的作用就是使 UI 看起来更加漂亮和流畅，最重要的是最后六行代码。我还定义了 UserControl 的高度和宽度，并将 u 赋值给 Popup 控件 p。（这里建议的宽度是 346 或者 646，高度应该是用户屏幕的完整高度，参考：设计指南）。最后，设置了 popup 的 left 和 top 属性，这样 popup 将会出现在适当的位置，然后将 p 返回给 SettingsCommand。

注解：Windows 8 可以根据机器的本地化设置而有不同的变化。如果有的国家的语言是从右往左读的，那么 Charms Bar 的实际位置是在屏幕的左边，而不是右边。这就是为什么我在给 popup 的 `LeftProperty` 赋值时，检查 SettingsPane 的“edge”。在 BuildSettingsItem 方法的开始处，你也可以看到类似的逻辑处理：PaneThemeTransition 的 dege。

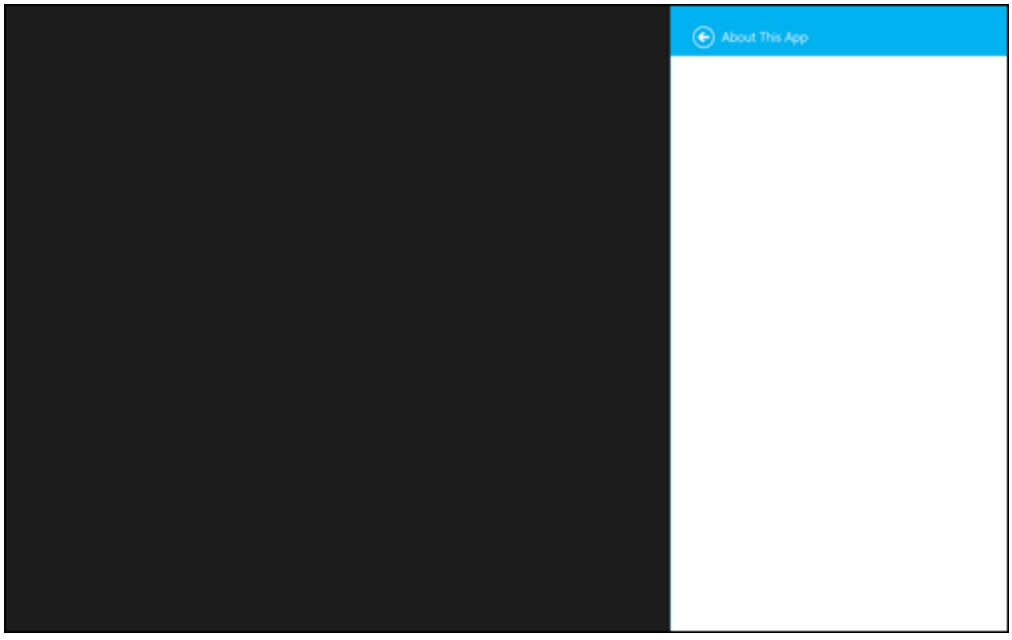
BuildSettingsItem 单独写了一个方法，是因为我认为当创建多个 SettingsCommand 时，会相对简单一些。

1.3. 创建 UserControl

我之前承诺过会介绍本节的内容，现在就开始哦。在这节，设计到的内容是设置合约所需要用到的。下面，我需要创建一个 page。添加一个新的 UserControl 项到工程中。为了跟之前的代码想匹配，我将其命名为 AboutPage.xaml，当然，你也可以使用任何你想用的名字。



这个文件中代码的内容完全取决于你。没有任何的官方文档约束。下面我提供的代码示例是一个简短的模板，代码实现的界面看起来是这样的：



你也可以使用任意的 xaml 代码来编写这个界面。

```
<Border BorderBrush="#00b2f0" BorderThickness="1,0,0,0">
<Grid Background="White" VerticalAlignment="Stretch">
<Grid.RowDefinitions>
<RowDefinition Height="Auto"/>
<RowDefinition Height="*" />
</Grid.RowDefinitions>

<!-- HEADER -->
<Grid Background="#00b2f0" Grid.Row="0">
<Grid.ColumnDefinitions>
<ColumnDefinition Width="*" />
</Grid.ColumnDefinitions>

<StackPanel Orientation="Horizontal" Grid.Column="0" Margin="40, 32, 17, 13">
<Button x:Name="BackButton" Margin="0,3,0,0" Style="{StaticResource BackButtonStyle}"
Click="BackButton_Click"/>
<TextBlock Margin="10,10,0,0" FontFamily="Segoe UI" FontWeight="SemiLight" FontSize="24.6667"
Text="About This App" Foreground="White"/>
</StackPanel>
<Image Source="Assets/SmallLogo.png" Width="29" Height="29" Margin="353,46,17,21" />
</Grid>

<!-- CONTENT AREA -->
<ScrollViewer VerticalScrollBarVisibility="Auto" Grid.Row="1">
<Grid Margin="40,33,40,39" VerticalAlignment="Top" Grid.RowSpan="3">
```

```
<StackPanel>

</StackPanel>
</Grid>
</ScrollViewer>
</Grid>
</Border>
```

上面的代码，没有大量复杂的布局。只有三个对于 header 比较重要的控件：

- Button——Button 按钮是一个圆形的箭头。这里实现了一个名为 BackButton_Click 的 click 事件，在本文的最后，我将介绍这个事件相关的代码。

- TextBlock——这个包含了设置相关的名称。大多数情况下，这里的内容应该与 SettingsPane 中出现的内容一样

- Image——这个图片不是必须的，当然，为了尽可能的展示程序，建议这里放置一个相关的图标。

在底部，是用来放置内容的区域。在嵌入的 StackPanel 控件中，可以布局你的内容数据。这里可以是任何内容，但是最好保持简单。CheckBox，ToggleButton，TextBox 和 RadioButton 等。简单的控件，让人看起来是执行一个简单的任务。

下面是 BackButton_Click 函数的代码：

```
privatevoid BackButton_Click(object sender, RoutedEventArgs e)
{
Popup parent = this.Parent asPopup;
if (parent != null)
      {
            parent.IsOpen = false;
```

```
        }

// If the app is not snapped, then the back button shows the Settings pane again.
if (Windows.UI.ViewManagement.ApplicationView.Value !=
Windows.UI.ViewManagement.ApplicationViewState.Snapped)
        {
SettingsPane.Show();
        }
}
```

上面的代码可以看到，我关闭了父控件——Popup 控件。如果程序不是在 snapped 状态，则重新打开 SettingsPane，这是 back 按钮应该做的。

1.4. 保存数据

噢，稍等。你可能想知道如何保存用户数据？

Windows 8 已经改变了许多东西，其中一个核心的用户行为概念已经消失了："保存"按钮。尽可能的，我希望你消除"保存"这个概念。

当用户修改了设置中的一个值：

别等到用户点击"保存"按钮才保存数据，

当操作是可逆的，不要弹出"确定？"对话框。

当数据改变的时候就进行保存，这样可以给用户提供更好的体验。如果用户点击一个"删除所有数据"选项时，可能需要提醒用户进行确认。但是这仅仅是当这个操作不可逆的才提醒。别把用户当做白痴。

至于如何保存数据，我将在第 8 日进行介绍。到时候我会花大量的时间来介绍将数据如何存储在设备本地，以及利用一个账号，将数据值漫游在不同的机器

上。

1.5. 总结

当我第一次接触 Windows 8 的时候，我经常听到这个词“Win as 1”。Win as 1 是 Windows 的核心原则之一。起初，这听起来像是营销者说的话，但实际上不是这样的。现在我已经在 Windows 8 上工作一段时间了，合约就是遵循了这一核心原则。想想，现在程序的设置都统一在一个地方呈现给用户，我们的程序可以将更多的注意力放在其功能上，而不是设置上面。甚至所有的程序设置都是一致的，我们只需要教会用户一次设置的使用，那么用户就能使用所有程序的设置了。

“只需要教会我打开一道门，我就能打开所有的门。”

今天，我们学习了设置合约，你会发现设置合约是 Windows 8 许多合约中的一个。就像一个编程接口，Windows 8 中的合约给你提供一个统一的方法来扩展程序，并且给用户提供了统一的用户界面。

可以点击下面的按钮下载完整的示例代码：

明天我将介绍搜索合约。不论是从可用性的角度来看，还是将程序放置在用户前面，搜索都非常的有价值。明天见。

感谢你的阅读！

如果对这篇文章有什么想法想法，可以与破船联系，破船的联系方式在文章开头。

破船