Windows 8 开发 31 日

第 04 日

新控件

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 04 日新控件 5

1.0. 介绍 5
1.1. AppBar 5
1.2. FliView 9
1.3. GridView 12
1.4. ProgressRing 17
1.5. ScrollViewer 18
1.6. SemanticZoom 19
1.7. WebView 28
1.8. 总结 28

第 04 日新控件

1.0. 介绍

今天我们来学习一下基础知识——XAML 程序中新增的布局控件。本文将介绍如下列出来的控件：

- AppBar
- FlipView
- GridView
- ProgressRing
- ScrollViewer
- SemanticZoom
- WebView

在这里的链接，可以看到 Windows 8 中 XAML 开发可用的全部控件。

1.1. AppBar

当我在写这一节的时候，我想起第一次写 ApplicationBar in Windows Phone development。在 Windows Phone 中非常的 AppBar 简单直接，但是你需要为每个按钮创建对应的图标，并且 Windows Phone OS 会为每个按钮添加一个圆圈。

当我开始了解 Windows 8 的 AppBar 时，发现不一定需要用图标，虽然可以很简单的实现带图标的功能。下面看看我将要创建的 AppBar 是什么样：

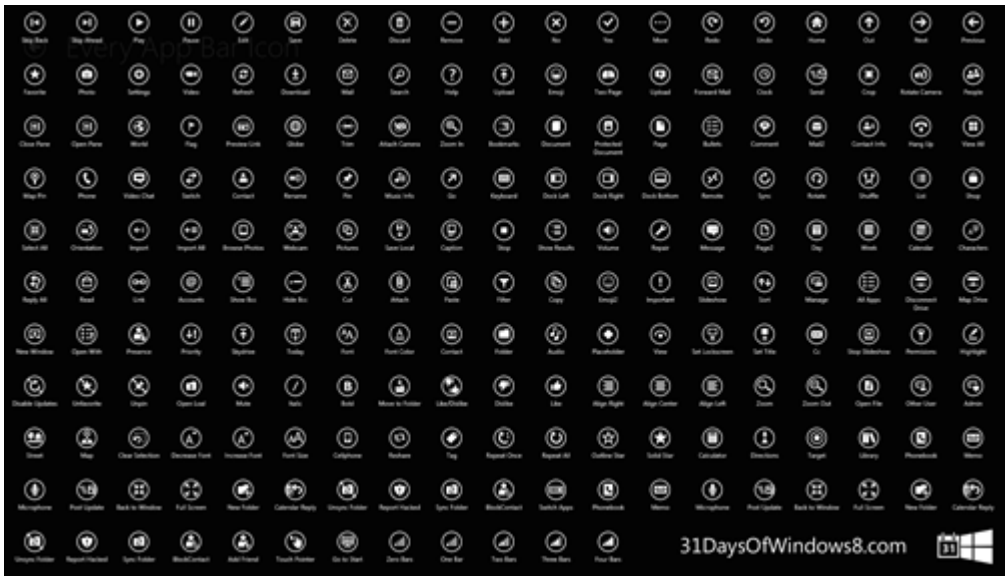
从上图中可以看到，我实现了 top 和 bottom AppBar。虽然从技术的角度上来讲，可以在 AppBar 上防止任何内容，但是有两条规则，需要注意：

1、 bottom AppBar 一般分为两部分。左边部分的 command 主要针对当前屏幕上下文。而右边的 command 则在整个程序中都通用，可以在程序的许多地方被访问。

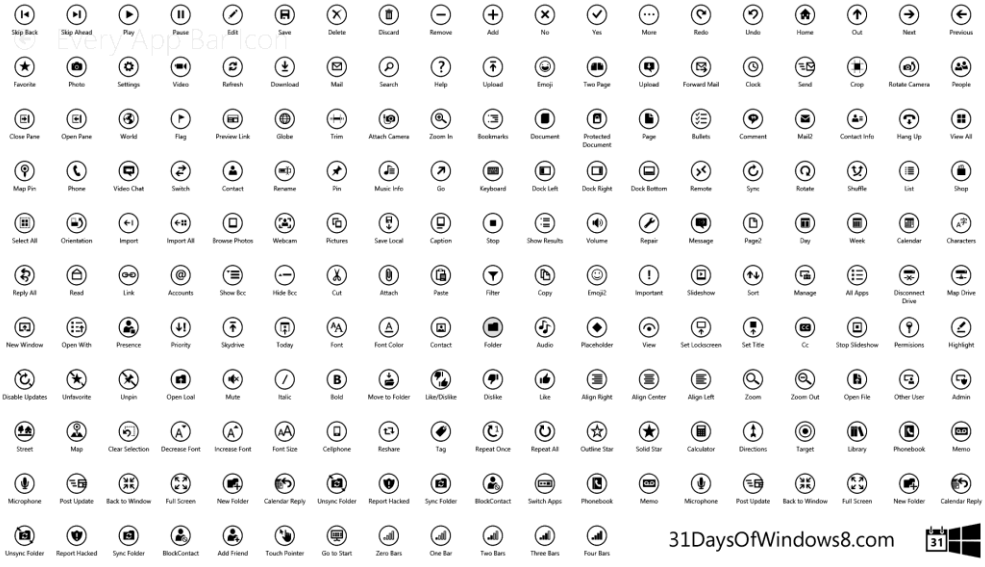
2、 top AppBar 则建议在程序中用来进行导航和其它一些选项操作，可以在官方的一些程序中发现这些规则。

微软已经为我们提供了 202 个默认的 AppBar styles，我们在创建 button 的时候直接使用即可。这 202 个 style 使用了“Segoe UI Icons”，并且这些 styles

内置于 StandardStyles.xaml 文件中。下面这个图显示了 StandardStyles.xaml 中所有默认可以使用的图标：



点击图片可以查看大尺寸的图。



好的，本节剩下的内容该来实现我的 AppBar 了。当用户进行手势操作时，为

了响应获得 AppBar，需要创建一个 Page.TopAppBar 或 Page.BottomAppBar，类似如下代码：

```
<Page.BottomAppBar>
<AppBar>
<!-- Put any content here that you'd like. -->
</AppBar>
</Page.BottomAppBar>
```

你可以在上面代码的注释部分添加任何内容，但是，如果你跟着规则走的话，你将需要把 AppBar 分为两部分。在这里，bottom bar 包含一个 Grid，这个 Grid 将 AppBar 平分为两部分，然后使用 StackPanels 来排放适当的图标。

```
<Page.BottomAppBar>
<AppBar>
<Grid>
<Grid.ColumnDefinitions>
<ColumnDefinition Width="50*" />
<ColumnDefinition Width="50*" />
</Grid.ColumnDefinitions>
<StackPanel Orientation="Horizontal"HorizontalAlignment="Left">
<Button x:Name="EditButton" Style="{StaticResourceEditAppBarButtonStyle}" Tag="Edit" />
</StackPanel>
<StackPanelGrid.Column="1" Orientation="Horizontal"HorizontalAlignment="Right">
<Button Style="{StaticResourceAddAppBarButtonStyle}" />
<Button Style="{StaticResourceHelpAppBarButtonStyle}" />
<Button Style="{StaticResourceDiscardAppBarButtonStyle}" />
<Button Style="{StaticResourceDeleteAppBarButtonStyle}" />
</StackPanel>
</Grid>
</AppBar>
</Page.BottomAppBar>
```

可以看到 bar 上的每一个图标，都是一个 Button 控件，每个 Button 控件使用了一种 style。之前说过，文件 StandardStyles.xaml 中默认的 202 种 style 是可以进行编辑的。默认情况下，所有的这些 style 都是被注释掉的，你可以对其取消注释，

或者只对用到的取消注释。

为了演示可以在 AppBar 上放置任何内容 我在页面的顶部放置了一些彩色块。

```
<Page.TopAppBar>
<AppBar>
<StackPanel Orientation="Horizontal" Margin="10,10,10,10" Height="200">
<Rectangle Width="200" Height="200" Fill="Red" Margin="0,0,20,0" Tapped="Rectangle_Tapped" />
<Rectangle Width="200" Height="200" Fill="Orange" Margin="0,0,20,0" Tapped="Rectangle_Tapped" />
<Rectangle Width="200" Height="200" Fill="Yellow" Margin="0,0,20,0" Tapped="Rectangle_Tapped" />
<Rectangle Width="200" Height="200" Fill="Green" Margin="0,0,20,0" Tapped="Rectangle_Tapped" />
<Rectangle Width="200" Height="200" Fill="Blue" Margin="0,0,20,0" Tapped="Rectangle_Tapped" />
<Rectangle Width="200" Height="200" Fill="Purple" Margin="0,0,20,0" Tapped="Rectangle_Tapped" />
</StackPanel>
</AppBar>
</Page.TopAppBar>
```

我在 StackPanel 中添加了一些 Rectangle 控件，并为每个 Rectangle 添加了 Tapped 事件。在 code-behind 文件中，首先获取被点击的 rectangle 的 Fill 颜色，然后将其设置为页面的背景色。方法如下：

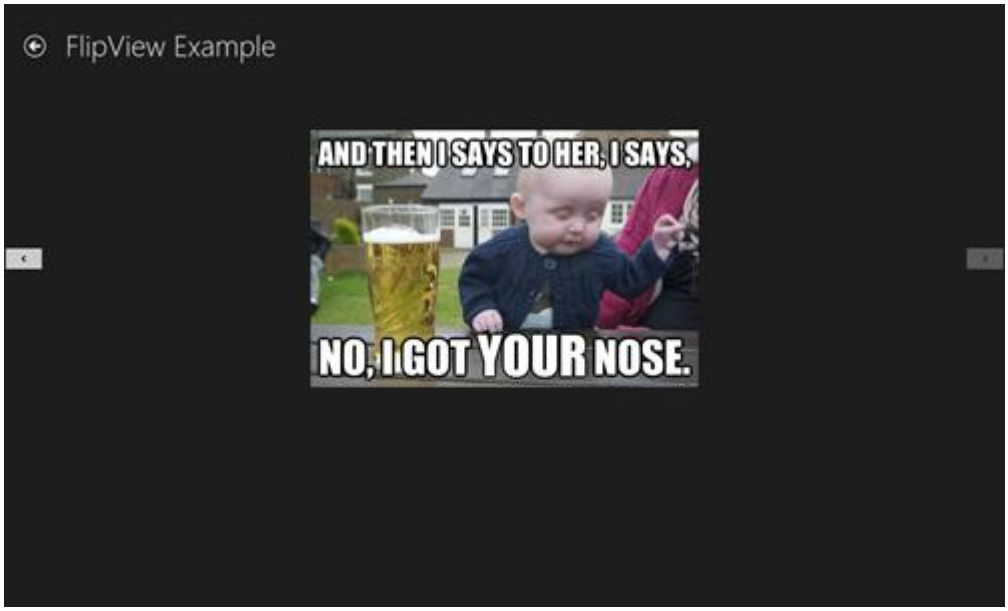
```
privatevoidRectangle_Tapped(object sender, TappedRoutedEventArgs e)
{
Rectanglerect = sender asRectangle;
LayoutRoot.Background = rect.Fill;
}
```

在本文的最后有链接，可以下载到示例代码。当运行代码时，请注意默认的 Windows 8 手势已经在工作了。只需要从屏幕的底部或者顶部轻扫，即可显示 AppBar，当然，在程序中，右键单击也可以显示出来。

1.2. FliView

FlipView 控件在 Windows Phone 中是没有的，我曾在 Windows Phone 中构建过。FlipView 控件允许用户可以翻转一组特定的内容。在这里的示例中，我将翻

转一组图片。由于 FlipView 是继承自 ItemsControl 的，所以，可以绑定任何类型的数据到该控件上，也可以使用 DataTemplates 进行复杂的布局。下面我们先浏览一下运行中的 FlipView 控件（点击看大尺寸）：



如上图所示，FlipView 控件提供了向前和向后的箭头，以此提醒当前图片在图片列表中的位置，并且还可以防止继续在开头和结尾处进行翻转。这个控件的使用方法非常的简单：

```
<FlipView x:Name="Flipper">
<Image Source="Assets/meme1.jpg" />
<Image Source="Assets/meme2.jpg" />
<Image Source="Assets/meme3.jpg" />
<Image Source="Assets/meme4.jpg" />
</FlipView>
```

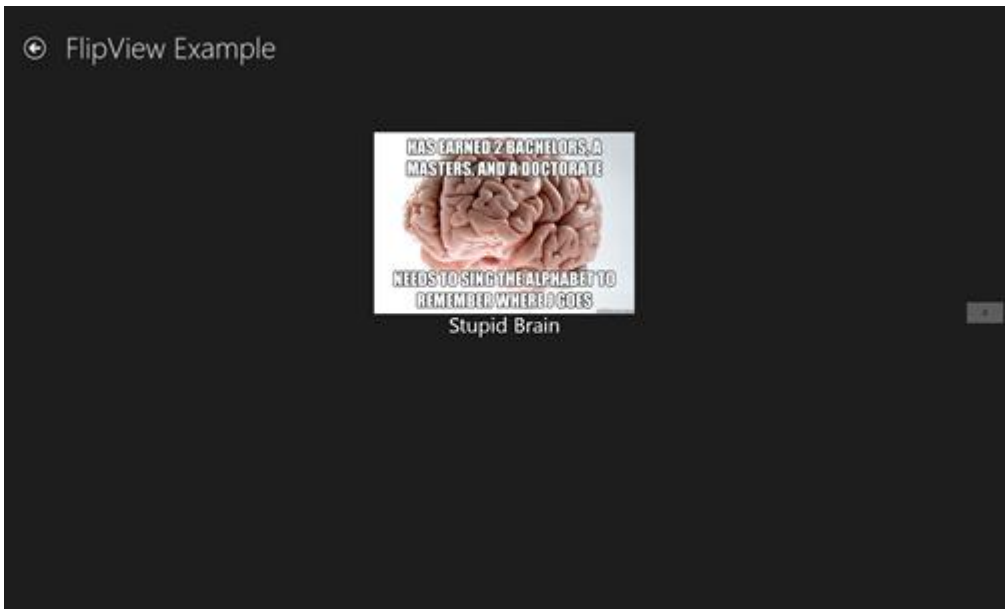
你唯一需要做的事情就是往 FlipView 控件中添加图片，这样就实现了一个简单的图片浏览器！如果你不满足于此，还有一些更复杂的需求，比如给图片浏览

器添加图片标题，那么可以使用 FlipViewItems，FlipViewItems 可以包吃住它自己的一个值，如下代码：

All you need to do is add images to a FlipView, and you've got a simple image browser ready to go! For something more complex, you might want to add captions to your image browser. To do this, we can use FlipViewItems, with each Item holding its own unique values, like this:

```
<FlipView x:Name="Flipper">
<FlipViewItem>
<StackPanel>
<Image Source="Assets/meme1.jpg" Width="500" />
<TextBlock Text="Stupid Brain"TextAlignment="Center"FontSize="40" />
</StackPanel>
</FlipViewItem>
<FlipViewItem>
<StackPanel>
<Image Source="Assets/meme2.jpg" Width="500" />
<TextBlock Text="Drunk Baby"TextAlignment="Center"FontSize="40" />
</StackPanel>
</FlipViewItem>
<FlipViewItem>
<StackPanel>
<Image Source="Assets/meme3.jpg" Width="500" />
<TextBlock Text="Link Baby"TextAlignment="Center"FontSize="40" />
</StackPanel>
</FlipViewItem>
<FlipViewItem>
<StackPanel>
<Image Source="Assets/meme4.jpg" Width="500" />
<TextBlock Text="Keanu Mario"TextAlignment="Center"FontSize="40" />
</StackPanel>
</FlipViewItem>
</FlipView>
```

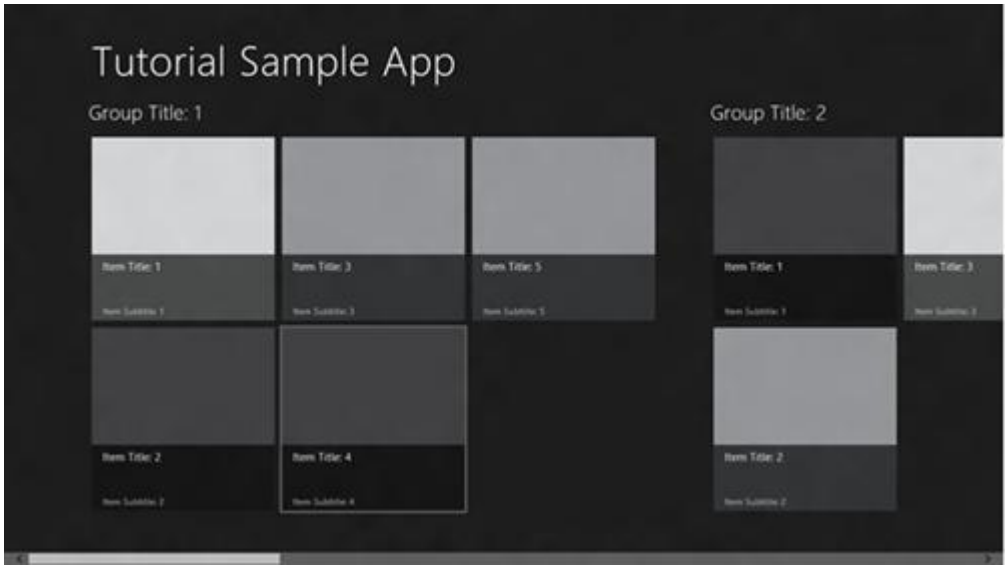
如上代码，可以为每一个图片添加一个标题。如下（单击它）：



这里有许多方法可以扩展这个控件，强烈建议你看一下 MSDN 上的例子：FlipView control sample。

1.3. GridView

GridView 控件在 Windows 8 程序中非常常见。在这里，几乎每个 demo/example 都会使用到该控件，在 Visual Studio 中，它也是 Grid Application template 的核心。



GridView 跟 FlipView 和 ListView 一样，都继承自 ItemsControl，继承自 ItemsControl 意味着他们将可以包含任意类型的 XAML 内容。在这里，我将在 XAML 中创建一个 GridView，并将其填充一些示例数据。首先，定义一个示例数据类——一个简单的歌手列表：

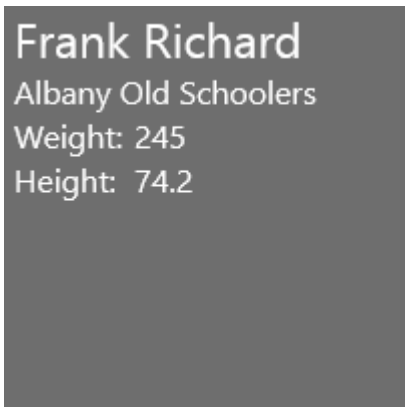
```
namespace Day4_NewControls
{
classPlayer
    {
publicstring Name { get; set; }
publicdouble Weight { get; set; }
publicint Number { get; set; }
publicdouble Height { get; set; }
publicstringTeamName { get; set; }
    }
}
```

如果你还没有用 Visual Studio 创建过一个类，那么请在项目名称上单击右键，选择“Add->Class…”或按 Shift+Alt+C。将类命名为 Player.cs，然后用上面的代码定义一个类。

下一步，创建 GridView 控件。首先，添加一个 GridView 到页面中：

```
<GridView x:Name="mainGridView" Background="#33FFFFFF">
</GridView>
```

要想将 GridView 控件做的更加的美观，那么还有一些工作要做。如果你读过我之前写的一篇关于 XAML 中数据绑定(点击这里查阅)的文章，你会知道任何继承自 ItemsControl 的控件都可以定义一个 ItemTemplate，该 ItemTemplate 可以指定每一个对象是如何显示的。在这里的示例中，我将使用每一个歌手为对象，并将它们在 GridView 中显示为一个矩形，并显示相关的数据。每一个矩形看起来如下图所示：



那么如何实现呢？通过定义一个 XAML 布局，该布局中包含我们想要显示的内容。下面是一个完整的 GridView，ItemTemplate 也在其中：

```
<GridView x:Name="mainGridView"Grid.Row="1" Background="#33FFFFFF"SelectionMode="Multiple">
<GridView.ItemTemplate>
<DataTemplate>
<StackPanel Height="200" Width="200" Background="#33FFFFFF">
<TextBlock Text="{Binding Name}"FontSize="24" Foreground="White" Margin="5" />
<TextBlock Text="{BindingTeamName}"FontSize="16" Foreground="White" Margin="5"/>
```

```
<StackPanel Orientation="Horizontal">
<TextBlock Text="Weight: "FontSize="16" Foreground="White" Margin="5"/>
<TextBlock Text="{Binding Weight}"FontSize="16" Foreground="White" Margin="5"/>
</StackPanel>
<StackPanel Orientation="Horizontal">
<TextBlock Text="Height:"FontSize="16" Foreground="White" Margin="5" />
<TextBlock Text="{Binding Height}"FontSize="16" Foreground="White" Margin="5" />
</StackPanel>
</StackPanel>
</DataTemplate>
</GridView.ItemTemplate>
</GridView>
```

上面可以看到，在绑定语句中，我已经将歌手的每个属性绑定好了，不过还需要创建对象才行，下面我通过代码，进行创建，打开 xaml.cs 文件：

在这里，我创建了一个 OnNavigatedTo 事件 handler，用这个函数来创建我的数据。首先我创建了一个 List<Player>集合，然后将该集合设置为 mainGridView 控件的 ItemsSource。

```
protectedoverridevoidOnNavigatedTo(NavigationEventArgs e)
{
base.OnNavigatedTo(e);

List<Player> players = newList<Player>();
players.Add(newPlayer { Name = "Frank Richard", TeamName = "Albany Old Schoolers", Height=74.2,
Weight=245, Number=03});
players.Add(newPlayer { Name = "Buddy Hobbs", TeamName = "New York Elves", Height = 76.8, Weight = 227,
Number = 04 });
players.Add(newPlayer { Name = "Ron Burgundy", TeamName = "San Diego Anchormen", Height = 77.1, Weight =
275, Number = 05 });
players.Add(newPlayer { Name = "Ricky Bobby", TeamName = "Talladega Racers", Height = 70.6, Weight = 198,
Number = 06 });
players.Add(newPlayer { Name = "Chazz Reinhold", TeamName = "Washington Crashers", Height = 70.6, Weight =
198, Number = 06 });

mainGridView.ItemsSource = players;
}
```

如果你编写正确，可以看到的界面是这样的：

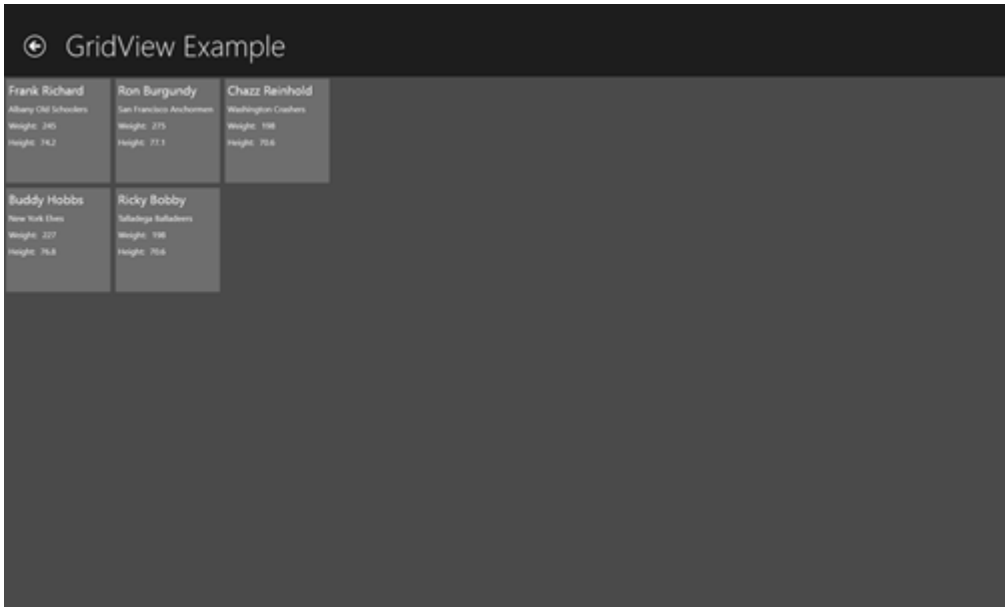
虽然这个界面是可以接受，但是当我开始写这个示例的时候，我希望的是每行有 3 个 items，并且有两行。为了实现这样的布局，可以编写大量的 xaml，或者看一下 GridView.ItemsPanel 吧。

重新定义一下 WrapGrid——可以定义行或者列的数目。在 GridView 控件中添加如下代码：

Redefining the WrapGrid that is there by default will allow you to define the number of rows or columns that will be used. Inside your GridView control, add these few lines:

```
<GridView.ItemsPanel>
<ItemsPanelTemplate>
<WrapGrid Orientation="Vertical"MaximumRowsOrColumns="2" />
</ItemsPanelTemplate>
</GridView.ItemsPanel>
```

然后运行程序，会看到类似如下界面：



1.4. ProgressRing

这个控件可能是本文中最简单的一个控件了，该控件也不是太重要。ProgressRing 控件被设计得非常易用，当在后台加载数据或者处理一些事情时，可以分散用户的注意力。将其放在画面中，起到视觉上的提醒，给用户的感觉是我这个程序还在运行着，让用户等待片刻。该控件看起来是这样的：

在页面中添加一个 ProgressRing 控件，只需要一行代码即可：

```
<ProgressRingx:Name="progressRing"IsActive="True"Visibility="Visible"Grid.Row="1"/>
```

在大多数情况下，你希望能够通过代码控制它。这也可以通过修改控件的 Visibility 属性即可：值为 Collapsed 或者 Visible。

我还发现一个很 cool 的事情，这个控件可以缩放到不同的尺寸。例如，下面这个画面中的 ProgessRing 尺寸是 400 x 400。

1.5. ScrollViewer

ScrollViewer 是另外一个相对简单的控件。你有多少次遇到这样的问题：你有大量的数据内容不能完全显示在一个屏幕上，而你必须虚构出一个 scrolling 或者 paging 来显示剩下的内容给用户。ScrollViewer 完全可以胜任你的问题。将你的数据内容放在 ScrollViewer 中，并可以定义你想看到的滚动条。下面的示例中有两个 StackPanels，都排满了图片，只有第二个在 ScrollViewer 中。

The ScrollViewer is another incredibly simple control that is practically invaluable. How many times have you had a large set of content that just wouldn't fit

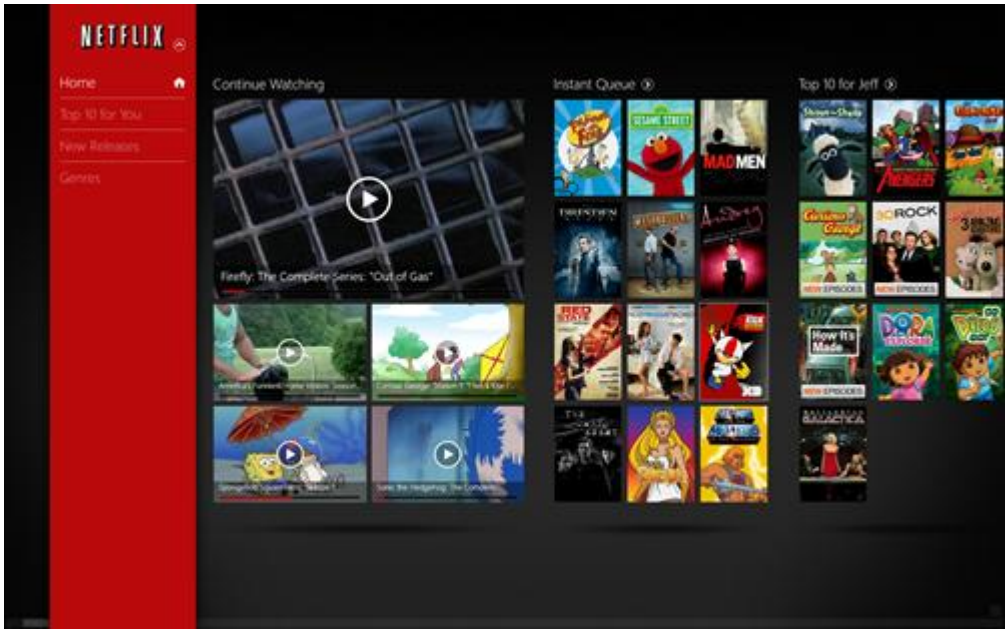
on the screen, and you had to invent a scrolling or paging mechanism to make the rest of the data available to the user? The ScrollViewer makes it simple. Stuff your content inside a ScrollViewer, define which scrollbars you want to see, and you're done. Here's an example that has two StackPanels full of images, except that only the second one is inside a ScrollViewer.

```
<StackPanel Orientation="Horizontal">
<Image Source="Assets/meme1.jpg" Height="300" />
<Image Source="Assets/meme2.jpg" Height="300" />
<Image Source="Assets/meme3.jpg" Height="300" />
<Image Source="Assets/meme4.jpg" Height="300" />
<Image Source="Assets/meme1.jpg" Height="300" />
<Image Source="Assets/meme2.jpg" Height="300" />
<Image Source="Assets/meme3.jpg" Height="300" />
<Image Source="Assets/meme4.jpg" Height="300" />
</StackPanel>
<ScrollViewerVerticalScrollBarVisibility="Hidden"HorizontalScrollBarVisibility="Auto">
<StackPanel Orientation="Horizontal">
<Image Source="Assets/meme1.jpg" Height="300" />
<Image Source="Assets/meme2.jpg" Height="300" />
<Image Source="Assets/meme3.jpg" Height="300" />
<Image Source="Assets/meme4.jpg" Height="300" />
<Image Source="Assets/meme1.jpg" Height="300" />
<Image Source="Assets/meme2.jpg" Height="300" />
<Image Source="Assets/meme3.jpg" Height="300" />
<Image Source="Assets/meme4.jpg" Height="300" />
</StackPanel>
</ScrollViewer>
```

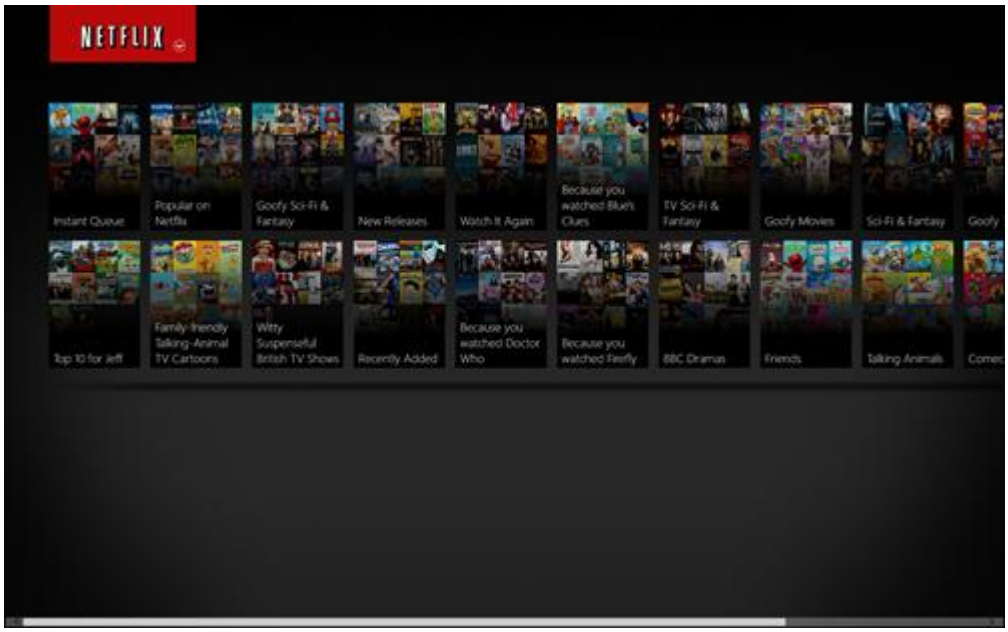
1.6. SemanticZoom

我第一次看到 SemanticZoom 是在 Windows 8 的开始屏幕上。Pinching 操作可以看到 apps 和 tile 的最小内容。(如果你的屏幕不支持触摸，可以使用 Ctrl+滚轮)。后来，我在许多程序中都看到了该控件，包括 Windows Store、News 和 Netflix。下面来看看 Netflix 程序。

Zoomed In(默认视图)：

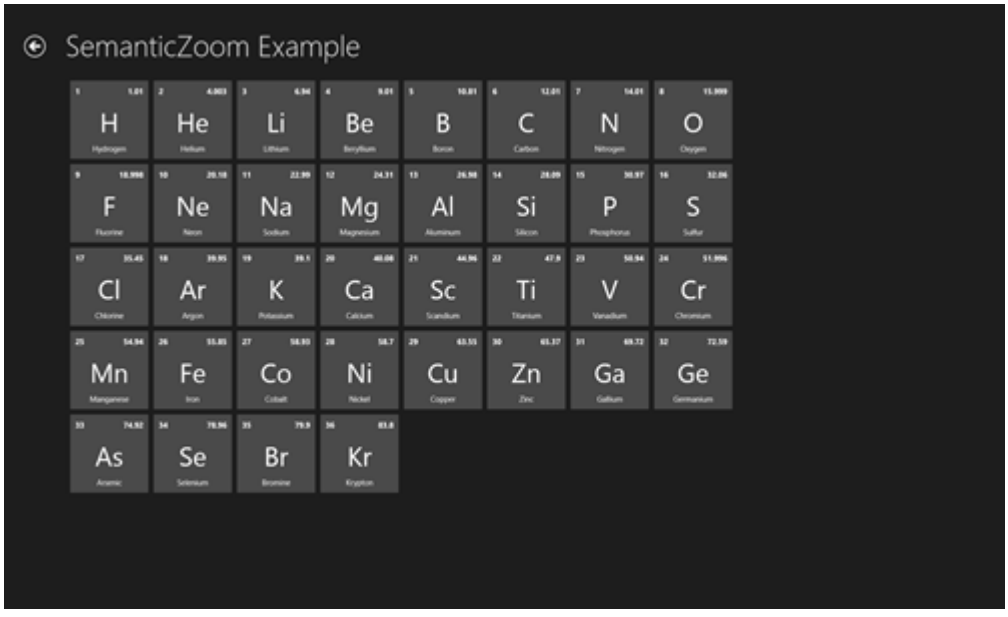


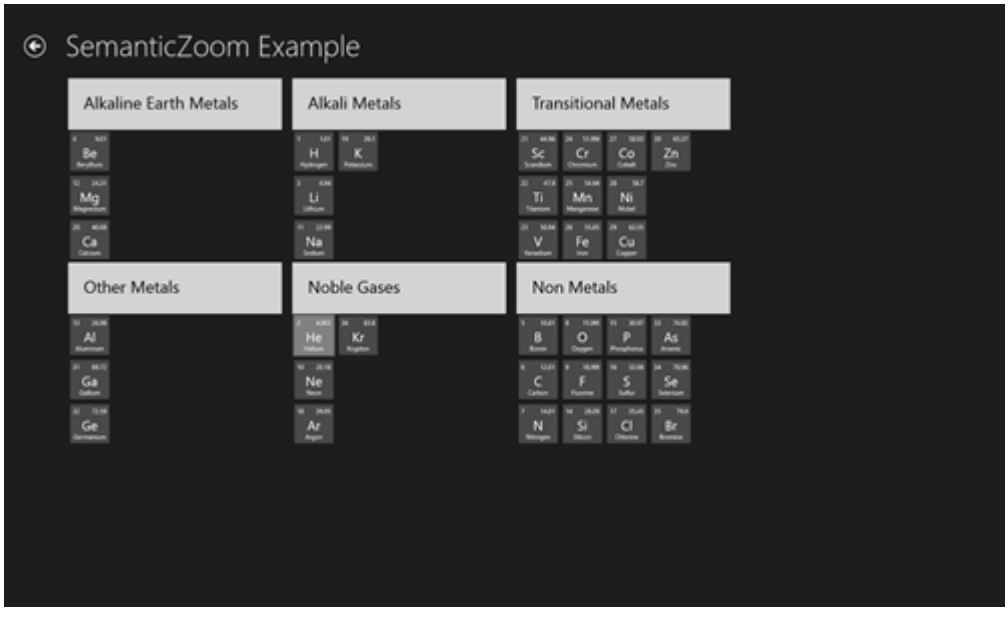
Zoomed Out (元视图):



SemanticZoom 的优点就是有两个视图，ZoomedOutView 和 ZoomedInView，理论上，在每个视图中我们可以放置我们想放的内容。（在微软的 MSDN 上，有一

个具体的指导：关于如何以及什么时候使用 SemanticZoom）。在这里，有一组元素周期表，我将显示元素列表，以及一个分组视图——该分组视图显示元素所在的家族，例如 Noble Gases 或 Alkali Metals，如下画面：





为了实现这个功能，需要从使用一个 SemanticZoom 控件。首先，添加一个

SemanticZoom 控件到页面中，如下：

```
<SemanticZoom>
<SemanticZoom.ZoomedInView>
<GridView>

</GridView>
</SemanticZoom.ZoomedInView>
<SemanticZoom.ZoomedOutView>
<GridView>

</GridView>
</SemanticZoom.ZoomedOutView>
</SemanticZoom>
```

如上代码，需要在这里添加一些内容。首先是两个节点：ZoomedInView 和 ZoomedOutView。我喜欢按照这样的顺序来排放，因为 ZoomedInView 是用户将看到的默认视图，而 ZoomedOutView 是缩放后看到的视图。这样很容易记住。

在这两个容器中，各需要一个实现了 ISemanticZoomInformation 接口的控件，目前只有 GridView 和 ListView 控件可以使用。因此，在本文中，我将使用 GridView。

在开始之前，我先说一下数据。本示例使用元素周期表中的元素，因此在这里需要一个 Element 类。

```
namespace Day4_NewControls
{
classElement
    {
publicdoubleAtomicWeight { get; set; }
publicintAtomicNumber { get; set; }
publicstring Symbol { get; set; }
publicstring Name { get; set; }
publicstring Category { get; set; }
publicstring State { get; set; }
    }
}
```

另外，还需要在页面中创建和加载一些数据。打开 Xaml.cs 文件，然后添加一

个 OnNavigatedTo 方法，如下代码：

```
protectedoverridevoidOnNavigatedTo(NavigationEventArgs e)
{
base.OnNavigatedTo(e);

List<Element> elements = newList<Element>();
elements.Add(newElement { AtomicNumber = 1, AtomicWeight = 1.01, Category = "Alkali Metals", Name =
"Hydrogen", Symbol = "H", State = "Gas" });
elements.Add(newElement { AtomicNumber = 2, AtomicWeight = 4.003, Category = "Noble Gases", Name =
"Helium", Symbol = "He", State = "Gas" });
elements.Add(newElement { AtomicNumber = 3, AtomicWeight = 6.94, Category = "Alkali Metals", Name =
"Lithium", Symbol = "Li", State = "Solid" });
elements.Add(newElement { AtomicNumber = 4, AtomicWeight = 9.01, Category = "Alkaline Earth Metals", Name
= "Beryllium", Symbol = "Be", State = "Solid" });
elements.Add(newElement { AtomicNumber = 5, AtomicWeight = 10.81, Category = "Non Metals", Name =
"Boron", Symbol = "B", State = "Solid" });
elements.Add(newElement { AtomicNumber = 6, AtomicWeight = 12.01, Category = "Non Metals", Name =
"Carbon", Symbol = "C", State = "Solid" });
elements.Add(newElement { AtomicNumber = 7, AtomicWeight = 14.01, Category = "Non Metals", Name =
"Nitrogen", Symbol = "N", State = "Gas" });
elements.Add(newElement { AtomicNumber = 8, AtomicWeight = 15.999, Category = "Non Metals", Name =
"Oxygen", Symbol = "O", State = "Gas" });
elements.Add(newElement { AtomicNumber = 9, AtomicWeight = 18.998, Category = "Non Metals", Name =
"Fluorine", Symbol = "F", State = "Gas" });
elements.Add(newElement { AtomicNumber = 10, AtomicWeight = 20.18, Category = "Noble Gases", Name =
"Neon", Symbol = "Ne", State = "Gas" });
elements.Add(newElement { AtomicNumber = 11, AtomicWeight = 22.99, Category = "Alkali Metals", Name =
"Sodium", Symbol = "Na", State = "Solid" });
elements.Add(newElement { AtomicNumber = 12, AtomicWeight = 24.31, Category = "Alkaline Earth Metals",
Name = "Magnesium", Symbol = "Mg", State = "Solid" });
elements.Add(newElement { AtomicNumber = 13, AtomicWeight = 26.98, Category = "Other Metals", Name =
"Aluminum", Symbol = "Al", State = "Solid" });
elements.Add(newElement { AtomicNumber = 14, AtomicWeight = 28.09, Category = "Non Metals", Name =
"Silicon", Symbol = "Si", State = "Solid" });
elements.Add(newElement { AtomicNumber = 15, AtomicWeight = 30.97, Category = "Non Metals", Name =
"Phosphorus", Symbol = "P", State = "Solid" });
elements.Add(newElement { AtomicNumber = 16, AtomicWeight = 32.06, Category = "Non Metals", Name =
"Sulfur", Symbol = "S", State = "Solid" });
elements.Add(newElement { AtomicNumber = 17, AtomicWeight = 35.45, Category = "Non Metals", Name =
"Chlorine", Symbol = "Cl", State = "Gas" });
elements.Add(newElement { AtomicNumber = 18, AtomicWeight = 39.95, Category = "Noble Gases", Name =
"Argon", Symbol = "Ar", State = "Gas" });
elements.Add(newElement { AtomicNumber = 19, AtomicWeight = 39.10, Category = "Alkali Metals", Name =
"Potassium", Symbol = "K", State = "Solid" });
elements.Add(newElement { AtomicNumber = 20, AtomicWeight = 40.08, Category = "Alkaline Earth Metals",
Name = "Calcium", Symbol = "Ca", State = "Solid" });
elements.Add(newElement { AtomicNumber = 21, AtomicWeight = 44.96, Category = "Transitional Metals", Name
= "Scandium", Symbol = "Sc", State = "Solid" });
elements.Add(newElement { AtomicNumber = 22, AtomicWeight = 47.90, Category = "Transitional Metals", Name
```

```
= "Titanium", Symbol = "Ti", State = "Solid" });
elements.Add(newElement { AtomicNumber = 23, AtomicWeight = 50.94, Category = "Transitional Metals", Name
= "Vanadium", Symbol = "V", State = "Solid" });
elements.Add(newElement { AtomicNumber = 24, AtomicWeight = 51.996, Category = "Transitional Metals", Name
= "Chromium", Symbol = "Cr", State = "Solid" });
elements.Add(newElement { AtomicNumber = 25, AtomicWeight = 54.94, Category = "Transitional Metals", Name
= "Manganese", Symbol = "Mn", State = "Solid" });
elements.Add(newElement { AtomicNumber = 26, AtomicWeight = 55.85, Category = "Transitional Metals", Name
= "Iron", Symbol = "Fe", State = "Solid" });
elements.Add(newElement { AtomicNumber = 27, AtomicWeight = 58.93, Category = "Transitional Metals", Name
= "Cobalt", Symbol = "Co", State = "Solid" });
elements.Add(newElement { AtomicNumber = 28, AtomicWeight = 58.70, Category = "Transitional Metals", Name
= "Nickel", Symbol = "Ni", State = "Solid" });
elements.Add(newElement { AtomicNumber = 29, AtomicWeight = 63.55, Category = "Transitional Metals", Name
= "Copper", Symbol = "Cu", State = "Solid" });
elements.Add(newElement { AtomicNumber = 30, AtomicWeight = 65.37, Category = "Transitional Metals", Name
= "Zinc", Symbol = "Zn", State = "Solid" });
elements.Add(newElement { AtomicNumber = 31, AtomicWeight = 69.72, Category = "Other Metals", Name =
"Gallium", Symbol = "Ga", State = "Solid" });
elements.Add(newElement { AtomicNumber = 32, AtomicWeight = 72.59, Category = "Other Metals", Name =
"Germanium", Symbol = "Ge", State = "Solid" });
elements.Add(newElement { AtomicNumber = 33, AtomicWeight = 74.92, Category = "Non Metals", Name =
"Arsenic", Symbol = "As", State = "Solid" });
elements.Add(newElement { AtomicNumber = 34, AtomicWeight = 78.96, Category = "Non Metals", Name =
"Selenium", Symbol = "Se", State = "Solid" });
elements.Add(newElement { AtomicNumber = 35, AtomicWeight = 79.90, Category = "Non Metals", Name =
"Bromine", Symbol = "Br", State = "Liquid" });
elements.Add(newElement { AtomicNumber = 36, AtomicWeight = 83.80, Category = "Noble Gases", Name =
"Krypton", Symbol = "Kr", State = "Gas" });

ElementData.Source = elements;
CategoryData.Source = from el in elements group el byel.Categoryintogrporderbygrp.Keyselectgrp;
}
```

这里，我添加了元素周期表中的前 36 个元素，如果你有时间的话，可以添加更多。现在，你可能看到了上面代码的最后两行涉及到的 ElementData 和 CategoryData。它们是 CollectionViewSource，我在 XAML page 的 Resources 节点中添加的，如下代码：

For this example, I've included the first 36 elements in the periodic table, but you can expand this to more if you've got the time. Right now, you're probably wondering about those last two lines referring to ElementData and CategoryData. These are CollectionViewSource elements that I've added to our XAML page's Resources section. Here's what my Page.Resources section looks like:

```
<Page.Resources>
<x:String x:Key="AppName">SemanticZoom Example</x:String>
<CollectionViewSource x:Name="ElementData" />
<CollectionViewSource x:Name="CategoryData"IsSourceGrouped="True"/>
</Page.Resources>
```

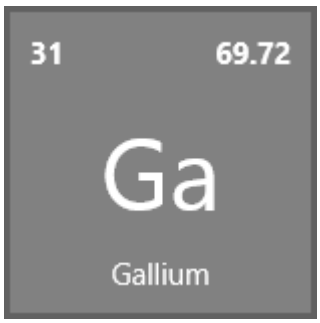
在 XAML 中定义数据的集合，上面是一种简单的方法，然后在代码中添加数据。它们分别代表两个不同版本的数据——分组和不分组。

下面我们先从 ZoomedInView 开始，它与之前介绍的 Grid 示例比较相似。在下面的代码中，我列出了完整的 ZoomedInView 代码，你可能已经注意到，我在 XAML 中声明了 ItemSource，而不是在后台代码中。我从 CollectionViewSource 中获取元数据。

```
<SemanticZoomGrid.Row="1" Margin="120,0,0,0">
<SemanticZoom.ZoomedInView>
<GridViewItemsSource="{Binding Source={StaticResourceElementData}}"SelectionMode="None">
<GridView.ItemTemplate>
<DataTemplate>
<Grid Width="150" Height="150" Background="#33FFFFFF">
<Grid.ColumnDefinitions>
<ColumnDefinition Width="10" />
<ColumnDefinition Width="65" />
<ColumnDefinition Width="65" />
<ColumnDefinition Width="10" />
</Grid.ColumnDefinitions>
<Grid.RowDefinitions>
<RowDefinition Height="10" />
<RowDefinition Height="35" />
<RowDefinition Height="60" />
<RowDefinition Height="35" />
<RowDefinition Height="10" />
</Grid.RowDefinitions>
<TextBlock Text="{BindingAtomicNumber}"Grid.Column="1"Grid.Row="1"FontWeight="Bold" />
<TextBlock
Text="{BindingAtomicWeight}"Grid.Column="2"Grid.Row="1"TextAlignment="Right"FontWeight="Bold" />
<TextBlock Text="{Binding
Symbol}"FontSize="50"Grid.Row="2"Grid.RowSpan="2"Grid.Column="1"Grid.ColumnSpan="2"TextAlignment="
Center" Width="130" />
<TextBlock Text="{Binding
Name}"Grid.Row="3"Grid.Column="1"Grid.ColumnSpan="2"TextAlignment="Center"VerticalAlignment="Bottom
" />
</Grid>
```

```
</DataTemplate>
</GridView.ItemTemplate>
<GridView.ItemsPanel>
<ItemsPanelTemplate>
<WrapGridMaximumRowsOrColumns="8" Orientation="Horizontal" />
</ItemsPanelTemplate>
</GridView.ItemsPanel>
</GridView>
</SemanticZoom.ZoomedInView>
```

剩余的代码是设计每一个元素的外观的，最终看起来如下：



上面就是关于 ZoomedInView。在图中，我选择 Gallium 作为截图，Gallium 非常好。它看起来像铝或者罐头，具有类似的重量。比较酷的是它的熔点是 86F 或 30C °。也就是说当你用手拿着的时候，它会融化，看起来更像汞。（感谢我的朋友 Jim Tocco 在多年直接向我介绍 Gallium）

下面我们接着看看 ZoomedOutView。在这里面，包含了更有趣的内容。下面的代码看起来几乎与 ZoomedInView 一样，只是字体和 Grid 更小一点。

```
<SemanticZoom.ZoomedOutView>
<GridView x:Name="GridIn"ItemsSource="{Binding
Source={StaticResourceCategoryData}}"SelectionMode="None">
<GridView.ItemTemplate>
<DataTemplate>
<Grid Width="75" Height="75" Background="#33FFFFFF">
<Grid.ColumnDefinitions>
<ColumnDefinition Width="5" />
<ColumnDefinition Width="33" />
<ColumnDefinition Width="32" />
<ColumnDefinition Width="5" />
```

```
</Grid.ColumnDefinitions>
<Grid.RowDefinitions>
<RowDefinition Height="5" />
<RowDefinition Height="18" />
<RowDefinition Height="30" />
<RowDefinition Height="17" />
<RowDefinition Height="5" />
</Grid.RowDefinitions>
<TextBlock Text="{BindingAtomicNumber}"Grid.Column="1"Grid.Row="1"FontSize="12" />
<TextBlock Text="{BindingAtomicWeight}"Grid.Column="2"Grid.Row="1"FontSize="12"TextAlignment="Right"
/>
<TextBlock Text="{Binding
Symbol}"FontSize="25"Grid.Row="2"Grid.Column="1"Grid.RowSpan="2"Grid.ColumnSpan="2"TextAlignment="
Center" Width="65" />
<TextBlock Text="{Binding
Name}"Grid.Row="3"Grid.Column="1"Grid.ColumnSpan="2"FontSize="12"TextAlignment="Center"VerticalAlign
ment="Bottom" />
</Grid>
</DataTemplate>
</GridView.ItemTemplate>
</GridView>
</SemanticZoom.ZoomedOutView>
```

下面，我再给 GridView 添加一个 GroupStyle——给每一个 section 添加一个 header。GroupStyle 定义了一个 HeaderTemplate，用来布局 header，并且绑定到 Category 名称。

下面是我写的 GridView.GroupStyle 标记代码。可以将其插入到上面代码的 GridViewItemTemplate 节点后面。

```
<GridView.GroupStyle>
<GroupStyleHidesIfEmpty="True">
<GroupStyle.HeaderTemplate>
<DataTemplate>
<StackPanel Background="LightGray" Margin="0,0,20,0">
<TextBlock Text="{Binding Key}" Foreground="Black" Margin="30"FontSize="32" Width="350" />
</StackPanel>
</DataTemplate>
</GroupStyle.HeaderTemplate>
<GroupStyle.Panel>
<ItemsPanelTemplate>
<VariableSizedWrapGrid/>
</ItemsPanelTemplate>
</GroupStyle.Panel>
```

```
</GroupStyle>
</GridView.GroupStyle>
```

GroupStyle 有它自己的 HeaderTemplate，可以定义每个 section 的 header。

现在，如果运行工程(或者再本文最后面下载示例)，你会发现在两个 SemanticZoom 视图之间可以很简单的进行过渡。

1.7. WebView

WebView 控件类似于 Windows Phone 开发中的 WebBrowser。简单的说，WebView 的作用是在程序中显示来自 web 或者本地文件中的 HTML 内容。

WebView 的使用也是简单的，只需要定义一个 source 属性就可以了：

```
<WebViewSource="http://31DaysOfWindows8.com"/>
```

1.8. 总结

这篇文章有点长，详细的介绍了 Windows 8 应用中使用 XAML 开发新增的 7 个控件。希望对你有一些作用。

点击下面的按钮，可以下载本文的示例代码和完整的解决方案。

明天，我将展开关于合约的一系列文章，在第 05 日，将介绍 Setting 合约。到时候见吧！

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