Understanding Pivot and Panorama

Part 4

Maybe you have at once interacted with a Windows Phone and have that wonderful user experience as you wipe between pages. Microsoft had come up with all these great features of Windows Phone and makes one have that natural feel of the left-to-right or right-to-left scrolling of apps and hubs. Yeah, all this had been made possible by the Microsoft UI controls called Pivot and Panorama.

The Windows Phone team looked into the behavior of how users interact with the phone and figured out that the anatomy of the hand is such that it's actually easier to scroll left and right when compared to up and down.

Basically pivot and panorama are like a tab control for desktop applications. These two controls are available in the Microsoft.Phone.Controls namespane from the assembly with the same name. They are similar controls allowing you to organize the visual content of your page into smaller parts (subpages or columns), which you can navigate through by sweeping the finger on the screen.

When coming up with app that uses any of this, panorama or pivot, one needs to understand that each "subpage", called item (pivot item or panorama item) has a header and you can also navigate through the items by taping on the header.

Both Pivot and Panorama controls are lightweight enough to have great performance, while visually engaging and responsive to touch. They share common features such as

- Beautiful easing animations for sliding during navigation
- Extensibility offered through events, visual states, and the ability to re-template
- Complete data binding support, content template properties and item container styles
- Automatic support for light and dark phone themes
- Simple XAML and application programming interfaces
- Standard items controls that feels natural to developers, individual items are built from PanoramaItem and PivotItem content controls
- Built-in touch navigation: flicks move quickly, while panning past a threshold and releasing the touch will snap to the next item
- Great tooling comes standard, including app and page templates

The most common similarity between Pivot and panorama is that both look like a big sheet or canvas grouped into subpages, and you have a view port into one of the subpages at a given time. But there's a wide difference between the two in the way the control title and the headers look and how the transition from one item to another occurs, the transition being smoother with panorama.

A. Panorama

Panorama control is a Windows Phone Control that can be used as the central application hub. You will agree with me that Panoramas are gorgeous, they leverage the Light, Clean, Open, Fast Metro Design Principle. Panorama are also designed to make the user feel like they are dealing with content that spans beyond the physical borders of the phone itself. The content peaking on the phone communicates continuity in a beautiful way tempting the user to swipe to the left or right and discover more content.

Unlike standard Windows Phone applications that are designed to fit within the confines of the phone screen, panorama applications offer a unique way to view controls, data, and services by using a long horizontal canvas that extends beyond the confines of the screen. Another thing is that Panorama allows the user of a Windows Phone to navigate into all of the areas of functionality in the app.

The structure of a Panorama contains a:

- Background image that is behind the whole control as its background,
- Home panel where you land when the app starts
- Additional panels that segment your UI at the top level of the app.

Each Panorama item has this three basic features:

- **Header** - this is the ContentControl that is used to display and animate the header.(Header is optional)
- **Content** - this is the ContentPresenter that displays the PanoramaItem content.
- **Orientation property** - setting the orientation to Horizontal will allow the item to be of a larger size than the control. By default, this property is Vertical, which means that panning or flicking should bring you to the next panorama section.

As you build your application, note the following:

- ✓ Use a properly sized image in your application. The recommended dimensions for a Pnorama background image is a height of 800 pixels and a width less than 2000 pixels.
- ✓ If a Panorama control is using an image for the background, its Build Action should be set to Resource; otherwise, it will not appear immediately when the application is first displayed. Setting the Build Action to Content would cause it to be loaded asynchronously.

When building the panorama, you will notice that, in your .xaml file, the Panorama control is placed on pages of your application and accepts a title and a background. Its sections are defined by PanoramaItem controls, and each has a header and a body. The Title and Header are easiest to use when you're just inserting text, but they also have associated HeaderTemplate and TitleTemplate dependency properties for data binding scenarios.

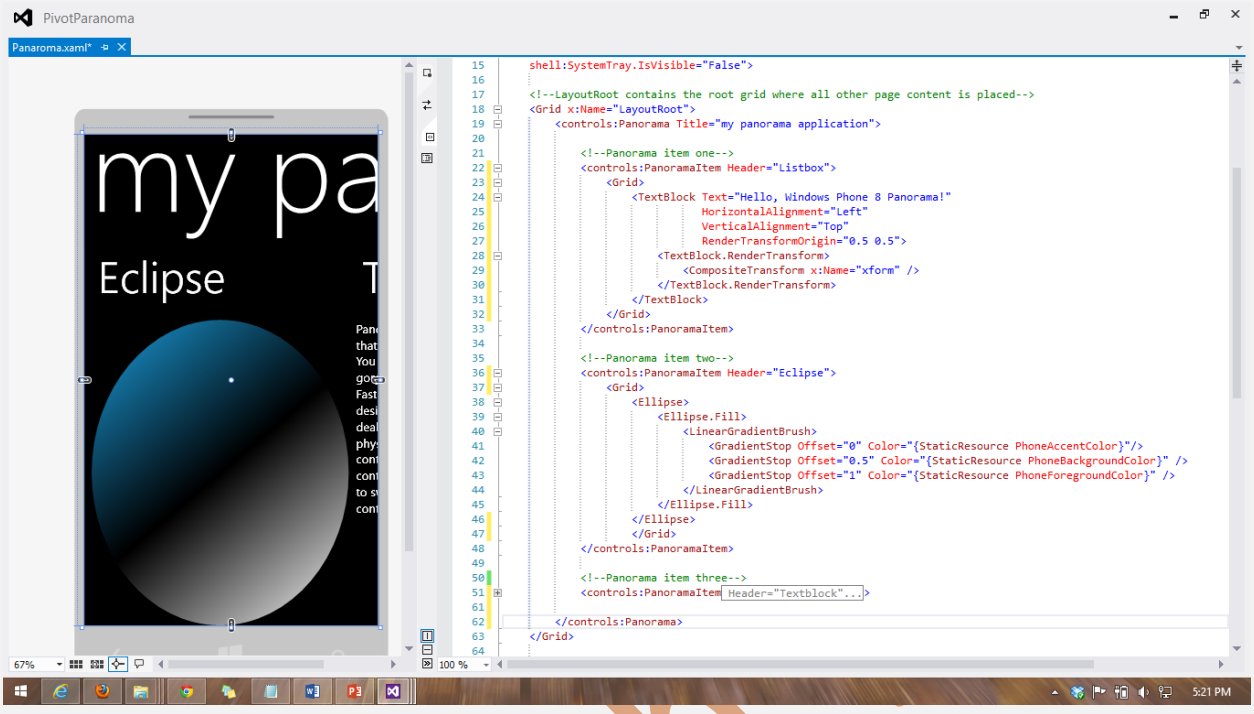
Let's now launch the Windows Phone SDK or your visual studio and create and new project and name it "PartFour". Create a new Page by Pressing 'Ctrl+Shift+A' and select Windows Phone Potrait Page and name it Panorama Page, on your xaml page, add the following pice of code.

"

```
<phone:PhoneApplicationPage
    x:Class="PartFour.PanoramaPage"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
                            xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
                            xmlns:phone="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
                            xmlns:shell="clr-
namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
                            xmlns:controls="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls"
                            xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
                            xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
                            mc:Ignorable="d"
                            d:DesignWidth="480"
                            d:DesignHeight="800"
                            FontFamily="{StaticResource PhoneFontFamilyNormal}"
                            FontSize="{StaticResource PhoneFontSizeNormal}"
                            Foreground="{StaticResource PhoneForegroundBrush}"
                            SupportedOrientations="Portrait"
                            Orientation="Portrait"
                            shell:SystemTray.IsVisible="False">

    <!--LayoutRoot contains the root grid where all other page content is placed-->
    <Grid x:Name="LayoutRoot">
        <controls:Panorama Title="my panorama application">

            <!--Panorama item one-->
            <controls:PanoramaItem Header="Listbox">
                <Grid>
                    <TextBlock Text="Hello, Windows Phone 8 Panorama!"
                               HorizontalAlignment="Left"
                               VerticalAlignment="Top"
                               RenderTransformOrigin="0.5 0.5">
                        <TextBlock.RenderTransform>
                            <CompositeTransform x:Name="xform" />
```

```
                    </TextBlock.RenderTransform>
                </TextBlock>
            </Grid>
        </controls:PanoramaItem>

        <!--Panorama item two-->
        <controls:PanoramaItem Header="Eclipse">
            <Grid>
                <Ellipse>
                    <Ellipse.Fill>
                        <LinearGradientBrush>
                            <GradientStop Offset="0"
                                          Color="{StaticResource PhoneAccentColor}"
/>
                            <GradientStop Offset="0.5"
                                          Color="{StaticResource
PhoneBackgroundColor}" />
                            <GradientStop Offset="1"
                                          Color="{StaticResource
PhoneForegroundColor}" />
                        </LinearGradientBrush>
                    </Ellipse.Fill>
                </Ellipse>
            </Grid>
        </controls:PanoramaItem>

        <!--Panorama item three-->
        <controls:PanoramaItem Header="Textblock">
            <Grid>
                <ScrollViewer>
                    <!-- from http://www.gutenberg.org/files/7178/7178-8.txt -->
                    <TextBlock TextWrapping="Wrap">
                        My text here
                    </TextBlock>
                </ScrollViewer>
            </Grid>
        </controls:PanoramaItem>

    </controls:Panorama>
</Grid>


</phone:PhoneApplicationPage>
”
```

As shown in the figure below.



Drag and drop a button on the MainPage.xaml Display layout and name it as 'panoramaButton', change its content to 'Panorama' double click it to create an event for you. And this code to the event, "`NavigationService.Navigate(new Uri("/PanoramaPage.xaml", UriKind.Relative));`".

Now run the application by pressing **'f5'!**

B. Pivot Pages

Pivot Pages are powerful, a horse power controls that have the ability to present and handle large amounts of data. They are virtualized so their performance is much better and optimized for handling large amounts of data compared to Panoramas. The Pivot control shares the left/right behavior of Panorama but the common experience to the two controls really ends there. The Pivot control is a deeper experience and meant to be data-driven. What is meant by this is the fact that Pivot controls are largely list-based and pivot along data points. In short pivots are best friends with ListBoxes (or ListView) controls that are used together in binding data.

Each Pivot Page have Pivot Items and each of these Pivots Items can present information from the same database but sorted out in different ways as you play around with them. For example in these two cases we have two different Pivots. The pivot control is a lot like a tab control, but it's designed specifically for the phone and touch interaction in that the entire pivot control can be panned, flicked, and manipulated. A pivot control provides a quick way to manage views or pages within the application. This control can be used for filtering large datasets, viewing multiple data sets, or switching application

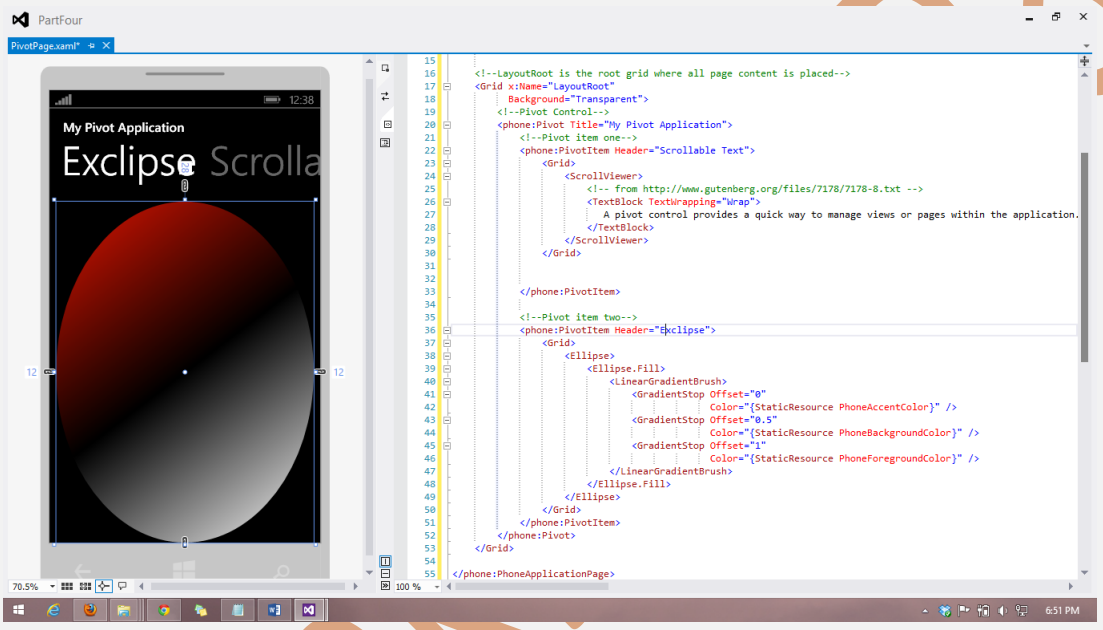
views. The control places individual views horizontally next to each other, and manages the left and right navigation. Flicking or panning horizontally on the page cycles the pivot functionality.

Let's get back to our PartFour Project and create an new "Windows Phone Pivot Page' and name it as 'PivotPage'. Add the following code on your 'PivotPage.xaml'.

"

```
<phone:PhoneApplicationPage
    x:Class="PartFour.PivotPage"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
                            xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
                            xmlns:phone="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
                            xmlns:shell="clr-
namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
                            xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
                            xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
                            mc:Ignorable="d"
                            FontFamily="{StaticResource PhoneFontFamilyNormal}"
                            FontSize="{StaticResource PhoneFontSizeNormal}"
                            Foreground="{StaticResource PhoneForegroundBrush}"
                            SupportedOrientations="Portrait"
                            Orientation="Portrait"
                            shell:SystemTray.IsVisible="True">

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot"
          Background="Transparent">
        <!--Pivot Control-->
        <phone:Pivot Title="My Pivot Application">
            <!--Pivot item one-->
            <phone:PivotItem Header="Eclipse">
                <Grid>
                    <ScrollViewer>
                        <!-- from http://www.gutenberg.org/files/7178/7178-8.txt -->
                        <TextBlock TextWrapping="Wrap">
                           My text here.
                        </TextBlock>
                    </ScrollViewer>
                </Grid>


            </phone:PivotItem>

            <!--Pivot item two-->
            <phone:PivotItem Header="Scrollable Text">
                <Grid>
                    <Ellipse>
                        <Ellipse.Fill>
                            <LinearGradientBrush>
                                <GradientStop Offset="0"
                                              Color="{StaticResource PhoneAccentColor}"
/>
                                <GradientStop Offset="0.5"
```

```
                                        Color="{StaticResource
PhoneBackgroundColor}" />
                                <GradientStop Offset="1"
                                        Color="{StaticResource
PhoneForegroundColor}" />
                            </LinearGradientBrush>
                        </Ellipse.Fill>
                    </Ellipse>
                </Grid>
            </phone:PivotItem>
        </phone:Pivot>
    </Grid>

</phone:PhoneApplicationPage>”
```



Drag and drop a button on the MainPage.xaml Display layout and name it as ‘pivotButton’, change its content to ‘Pivot’ double click it to create an event for you. And this code to the event, “`NavigationService.Navigate(new Uri("/PivotPage.xaml", UriKind.Relative));`”.

Now run the application by pressing **‘f5’!**

Source Code.

Kamiri Peterson M,
Game & Software Developer, Tutor & Mentor.
Google+: http://gplus.to/pbosoa
Blog: www.bhakitah.blogspot.com
Quick Links: www.kapes.yolasite.com
Nokia Store Links: Nokia Store
Windows Store Links: Windows Store