Windows® Phone

Hands-On Lab

Live Tile for WP7

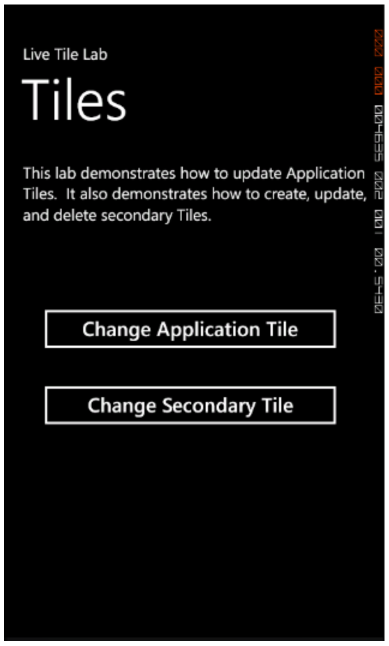
Live Tile for WP7

This sample HOL will demonstrate to you how to update an Application Tile using the ***ShellTile*** APIs. It also demonstrates how to create, delete, and update secondary Tiles.

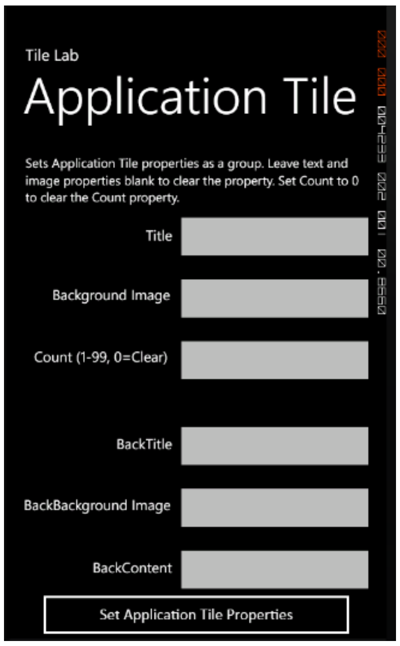
The lab contains three pages. The **MainPage** simply allows navigation to the two other pages. The **ApplicationTile** page allows you to update the properties of the Application Tile. All the properties are set at once. The **SecondaryTile** page allows you to create, update, and delete a secondary Tile. It also allows you to set each property individually and to demonstrate getting a value from the **QueryString**.

If the user navigates to the **SecondaryTile** page from the **MainPage**, the **DefaultTitle** value will be **FromMain**. If the user navigates to the **SecondaryTile** page by tapping the secondary Tile, the **DefaultTitle** value will be **FromTile**.

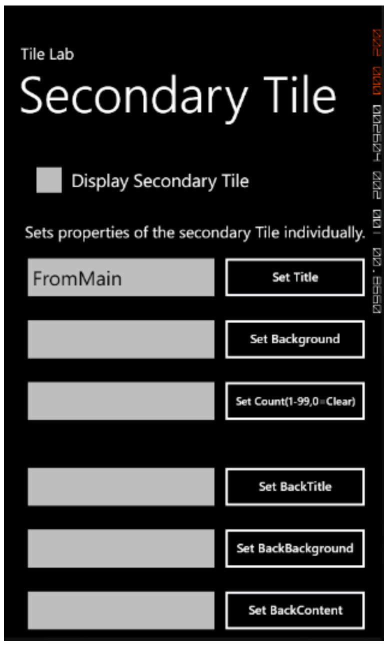
The UI for the 3 pages are as follows:



This lab demonstrates how to update Application Tiles. It also demonstrates how to create, update, and delete secondary Tiles.



Sets Application Tile properties as a group. Leave text and image properties blank to clear the property. Set Count to 0 to clear the Count property.



Sets properties of the secondary Tile individually.

FromMain

1. In Visual Studio 2010 Express for Windows Phone, create a new project by selecting the **File | New Project** menu command.
2. The **New Project** window is displayed. Expand the **Visual C#** templates, and then select the **Silverlight for Windows Phone** templates.
3. Select the **Windows Phone Application** template. Fill in the **Name** with **LiveTileLab**.
4. Click **OK**. The **New Windows Phone Application** window is displayed.
5. In the **Target Windows Phone Version** menu, ensure that **Windows Phone 7.1** is selected.
6. Click **OK**. A new project is created, and **MainPage.xaml** is opened in the Visual Studio designer window.
7. The instructor will be providing you 3 background images namely: **Red.jpg**, **Blue.jpg**, and **Green.jpg**. Add the three image files to the project by selecting the **LiveTileLab** project in **Solution Explorer** and selecting **Project | Add Existing Item** from the menu. Select the three image files and click **Add**. For each image file, set the **Build Action** property to **Content**.
8. Set the name of the application and page title in **MainPage** by replacing the **TitlePanel** with the XAML code below:

```
<!--TitlePanel contains the name of the application and page title-->
<StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
    <TextBlock x:Name="ApplicationTitle" Text="Live Tile Lab"
               Style="{StaticResource PhoneTextNormalStyle}"/>
    <TextBlock x:Name="PageTitle" Text="Tiles" Margin="9,-7,0,0"
               Style="{StaticResource PhoneTextTitle1Style}"/>
</StackPanel>
```

9. The XAML code for the **MainPage** is below. It contains one text block and two buttons. Each button will handle the **button_Click** event:

   Note: Replace the **Text** property value with: ***"This lab demonstrates how to update Application Tiles. It also demonstrates how to create, update, and delete secondary Tiles."***

```
        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <TextBlock Height="170"
                       HorizontalAlignment="Left"
                       Margin="12,6,0,0"
                       Name="textBlockDescription"
                       Text="….. "
                       VerticalAlignment="Top"
                       Width="438"
                       TextWrapping="Wrap"
                       TextAlignment="Left" />
            <Button Content="Change Application Tile"
                    Height="72"
                    HorizontalAlignment="Left"
                    Margin="28,182,0,0"
                    Name="buttonChangeApplicationTile"
                    VerticalAlignment="Top"
                    Width="392"
                    Click="buttonChangeApplicationTile_Click" />
            <Button Content="Change Secondary Tile"
                    Height="72"
                    HorizontalAlignment="Left"
                    Margin="28,278,0,0"
                    Name="buttonChangeSecondaryTile"
                    VerticalAlignment="Top"
                    Width="392"
                    Click="buttonChangeSecondaryTile_Click" />
        </Grid>
```

10. The next step is to create the UI of **ApplicationTile.xaml**. To create the **ApplicationTile** UI, select the **LiveTileLab** project name in the **Solution Explorer**, then select **Project | Add New Item** from the menu. Add a new **Windows Phone Portrait Page** and name it **ApplicationTile.xaml**. Click **Add**, and the **ApplicationTile.xaml** and **ApplicationTile.xaml.cs** files are created.

11. Set the name of the application and page title of the **ApplicationTile** page by replacing the **TitlePanel** with the XAML code below:

```
        <!--TitlePanel contains the name of the application and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle" Text="Tile Lab"
                       Style="{StaticResource PhoneTextNormalStyle}"/>
            <TextBlock x:Name="PageTitle" Text="Application Tile" Margin="9,-7,0,0"
                       Style="{StaticResource PhoneTextTitle1Style}"
                       FontSize="64" />
        </StackPanel>
```

12. The XAML code for the **ApplicationTile** page is below. It contains one text block, a series of six text blocks and six text boxes, and a button to set the properties. The **buttonSetApplicationTile** will handle the **button_Click** event:

```
        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <TextBlock Height="75" HorizontalAlignment="Left" Margin="12,6,0,0"
Name="textBlockDescription" Text="Sets Application Tile properties as a group. Leave text and
image properties blank to clear the property. Set Count to 0 to clear the Count property."
VerticalAlignment="Top" Width="438" TextWrapping="Wrap" FontSize="16" />
            <TextBlock Height="30" HorizontalAlignment="Left" Margin="164,96,0,0"
Name="textBlockTitle" Text="Title" VerticalAlignment="Top" FontSize="18" />
            <TextBox Height="72" HorizontalAlignment="Left" Margin="197,74,0,0"
Name="textBoxTitle" Text="" VerticalAlignment="Top" Width="260" />
            <TextBlock Height="30" HorizontalAlignment="Left" Margin="46,171,0,0"
Name="textBlockBackgroundImage" Text="Background Image" VerticalAlignment="Top" FontSize="18"
/>
            <TextBox Height="72" HorizontalAlignment="Left" Margin="197,152,0,0"
Name="textBoxBackgroundImage" Text="" VerticalAlignment="Top" Width="260" />
            <TextBlock Height="30" HorizontalAlignment="Left" Margin="23,249,0,0"
Name="textBlockCount" Text="Count (1-99, 0=Clear)" VerticalAlignment="Top" FontSize="18"
Width="175" />
            <TextBox Height="72" HorizontalAlignment="Left" Margin="197,230,0,0"
Name="textBoxCount" Text="" VerticalAlignment="Top" Width="260" InputScope="Number" />
            <TextBlock Height="30" HorizontalAlignment="Left" Margin="127,359,0,0"
Name="textBlockBackTitle" Text="BackTitle" VerticalAlignment="Top" FontSize="18" />
            <TextBox Height="72" HorizontalAlignment="Left" Margin="197,338,0,0"
Name="textBoxBackTitle" Text="" VerticalAlignment="Top" Width="260" />
            <TextBlock Height="30" HorizontalAlignment="Left" Margin="10,436,0,0"
Name="textBlockBackBackgroundImage" Text="BackBackground Image" VerticalAlignment="Top"
FontSize="18" />
            <TextBox Height="72" HorizontalAlignment="Left" Margin="197,416,0,0"
Name="textBoxBackBackgroundImage" Text="" VerticalAlignment="Top" Width="260" />
            <TextBlock Height="30" HorizontalAlignment="Left" Margin="97,515,0,0"
Name="textBlockContent" Text="BackContent" VerticalAlignment="Top" FontSize="18" />
            <TextBox Height="72" HorizontalAlignment="Left" Margin="197,494,0,0"
Name="textBoxBackContent" Text="" VerticalAlignment="Top" Width="260" />
            <Button Content="Set Application Tile Properties" Height="72"
HorizontalAlignment="Left" Margin="23,551,23,0" Name="buttonSetApplicationTile"
VerticalAlignment="Top" Width="410" FontSize="20" FontFamily="Calibri"
Click="buttonSetApplicationTile_Click" />
        </Grid>
```

13. The final step of creating the UI is to add the **SecondaryTile.xaml** page. To create the **SecondaryTile** UI, select the **LiveTileLab** project name in the **Solution Explorer**, then select **Project | Add New Item** from the menu. Add a new **Windows Phone Portrait Page** and name it **SecondaryTile.xaml**. Click **Add**, and the **SecondaryTile.xaml** and **SecondaryTile.xaml.cs** files are created.

14. Set the name of the application and page title of the **SecondaryTile** page by replacing the **TitlePanel** with the XAML code below:

```
        <!--TitlePanel contains the name of the application and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle" Text="Tile Lab" Style="{StaticResource
PhoneTextNormalStyle}"/>
            <TextBlock x:Name="PageTitle" Text="Secondary Tile" Margin="9,-7,0,0"
Style="{StaticResource PhoneTextTitle1Style}" FontSize="64" />
        </StackPanel>
```

15. The XAML code for the **SecondaryTile** page is below. It contains one check box, one text block, and a series of six text boxes and six buttons. The **checkBoxDisplaySecondaryTile** control will handle the **checkboxDisplaySecondaryTile_Checked** and **checkboxDisplaySecondaryTile_Unchecked** events. Each button will handle the **button_Click** events.

```
<!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <CheckBox Content="Display Secondary Tile " Height="72" HorizontalAlignment="Left"
Margin="20,4,0,0" Name="checkBoxDisplaySecondaryTile" VerticalAlignment="Top"
Checked="checkBoxDisplaySecondaryTile_Checked" Unchecked="checkBoxDisplaySecondaryTile_Unchecked"
/>
            <TextBlock Height="33" HorizontalAlignment="Center" Margin="9,91,0,0"
Name="textBlockDescription" Text="Sets properties of the secondary Tile individually."
VerticalAlignment="Top" Width="441" TextAlignment="Center" />

            <TextBox Height="72" HorizontalAlignment="Left" Margin="9,127,0,0"
Name="textBoxTitle" Text="" VerticalAlignment="Top" Width="260" />
            <Button Content="Set Title" Height="72" HorizontalAlignment="Left"
Margin="259,127,0,0" Name="buttonSetTitle" VerticalAlignment="Top" Width="200"
Click="buttonSetTitle_Click" FontSize="16" />

            <TextBox Height="72" HorizontalAlignment="Left" Margin="9,205,0,0"
Name="textBoxBackgroundImage" Text="" VerticalAlignment="Top" Width="260" />
            <Button Content="Set Background" Height="72" HorizontalAlignment="Left"
Margin="259,205,0,0" Name="buttonSetBackgroundImage" VerticalAlignment="Top" Width="200"
FontSize="16" Click="buttonSetBackgroundImage_Click" />

            <TextBox Height="72" HorizontalAlignment="Left" Margin="9,283,0,0"
Name="textBoxCount" Text="" VerticalAlignment="Top" Width="260"  InputScope="Number"/>
            <Button Content="Set Count(1-99,0=Clear)" Height="72" HorizontalAlignment="Left"
Margin="259,283,0,0" Name="buttonSetCount" VerticalAlignment="Top" Width="200"
Click="buttonSetCount_Click" FontSize="14" />

            <TextBox Height="72" HorizontalAlignment="Left" Margin="9,391,0,0"
Name="textBoxBackTitle" Text="" VerticalAlignment="Top" Width="260" />
            <Button Content="Set BackTitle" Height="72" HorizontalAlignment="Left"
Margin="259,391,0,0" Name="buttonSetBackTitle" VerticalAlignment="Top" Width="200"
Click="buttonSetBackTitle_Click" FontSize="16" />

            <TextBox Height="72" HorizontalAlignment="Left" Margin="9,469,0,0"
Name="textBoxBackBackgroundImage" Text="" VerticalAlignment="Top" Width="260" />
            <Button Content="Set BackBackground" Height="72" HorizontalAlignment="Left"
Margin="259,469,0,0" Name="buttonSetBackBackgroundImage" VerticalAlignment="Top" Width="200"
FontSize="16" Click="buttonSetBackBackgroundImage_Click" />

            <TextBox Height="72" HorizontalAlignment="Left" Margin="9,547,0,0"
Name="textBoxBackContent" Text="" VerticalAlignment="Top" Width="260" />
            <Button Content="Set BackContent" Height="72" HorizontalAlignment="Left"
Margin="259,547,0,0" Name="buttonSetBackContent" VerticalAlignment="Top" Width="200"
FontSize="16" Click="buttonSetBackContent_Click" />
        </Grid>
```

16. Implementing **MainPage** requires adding event handlers for the two buttons on **MainPage**.
17. Add the following code **to MainPage.xaml.cs**. This code implements the event handlers for the **button_Click** event. When a button is clicked, the application navigates to the appropriate page. When navigating to the **SecondaryTile** page, the parameter **DefaultTitle** is set to **FromMain** so that the **SecondaryTile** page knows where navigation came from. Later, we'll set **DefaultTitle** to **FromTile** when navigating to the **SecondaryPage** from a Tile.

```
        private void buttonChangeApplicationTile_Click(object sender, RoutedEventArgs e)
        {
            this.NavigationService.Navigate(new Uri("/ApplicationTile.xaml",
UriKind.Relative));
        }

        private void buttonChangeSecondaryTile_Click(object sender, RoutedEventArgs e)
        {
            this.NavigationService.Navigate(new
Uri("/SecondaryTile.xaml?DefaultTitle=FromMain", UriKind.Relative));
        }
```

18. To test the application at this point, press F5 to start debugging. The application will start in the emulator. Click one of the buttons. You should navigate to the appropriate page. Press the **Back** button on the phone emulator to return to the **MainPage** and test the other button. Press **Shift+F5** in Visual Studio to stop debugging.
19. Implementing the **ApplicationTile** page will require adding an event handler for the **buttonSetApplicationTile_Click** event.
20. Add a using directive to the top of the **ApplicationTile.xaml.cs** file for the **ShellTile** namespace.

```
//Import the ShellTile namespace
using Microsoft.Phone.Shell;
```

21. Add the following code to **ApplicationTile.xaml.cs**. This code will implement the event handler for the **buttonSetApplicationTile_Click** event. It will take the entered values in the text boxes and set all the properties of the Application Tile at once.

```
               int newCount = 0;

              // Application Tile is always the first Tile, even if it is not pinned to
Start.
              ShellTile TileToFind = ShellTile.ActiveTiles.First();

              // Application should always be found
              if (TileToFind != null)
              {
                  // if Count was not entered, then assume a value of 0
                  if (textBoxCount.Text == "")
                  {
                      // A value of '0' means do not display the Count.
                      newCount = 0;
                  }
                  // otherwise, get the numerical value for Count
                  else
                  {
                      newCount = int.Parse(textBoxCount.Text);
                  }

                  // Set the properties to update for the Application Tile.
                  // Empty strings for the text values and URIs will result in the property
being cleared.
                  StandardTileData NewTileData = new StandardTileData
                  {
                      Title = textBoxTitle.Text,
                      BackgroundImage = new Uri(textBoxBackgroundImage.Text,
UriKind.Relative),
                      Count = newCount,
                      BackTitle = textBoxBackTitle.Text,
                      BackBackgroundImage = new Uri(textBoxBackBackgroundImage.Text,
UriKind.Relative),
                      BackContent = textBoxBackContent.Text
                  };

                  // Update the Application Tile
                  TileToFind.Update(NewTileData);
              }
```

22. To test the application at this point, press F5 to start debugging. The application will start in the emulator. Click the **Change Application Tile** button. Set some properties for the Application Tile and click the **Set Application Tile Properties** button.

23. Add a using directive to the top of the **SecondaryTile.xaml.cs** file for the *ShellTile* namespace.

```
//Import the ShellTile namespace
using Microsoft.Phone.Shell;
```

24. Add the **OnNavigatedTo** event handler. Whenever we navigate to this page, we need to check to see whether the secondary Tile exists and set the check box appropriately. The default value for the **Title** is set here also, depending on the **DefaultTitle** set in the **QueryString**.

```
        protected override void OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e)
         {
             base.OnNavigatedTo(e);

             // See whether the Tile is pinned, and if so, make sure the check box for it is
checked.
             // (User may have deleted it manually.)
             ShellTile TileToFind = ShellTile.ActiveTiles.FirstOrDefault(x =>
x.NavigationUri.ToString().Contains("DefaultTitle=FromTile"));

             checkBoxDisplaySecondaryTile.IsChecked = (TileToFind != null);

             // To demonstrate the use of the Navigation URI and query parameters, we set the
default value
             // of the Title text box based on where we navigated from. If we navigated to
this page
             // from the MainPage, the DefaultTitle parameter will be "FromMain". If we
navigated here
             // when the secondary Tile was tapped, the parameter will be "FromTile".
             textBoxTitle.Text = this.NavigationContext.QueryString["DefaultTitle"];

         }
```

25. Add the following code for the **checkBoxDisplaySecondaryTile_Checked** event handler. This code checks to see whether a Tile with our designated ID already exists. If the Tile does not exist, the code creates a *StandardTileData* object and uses it to set the properties of the front and back of the Tile.

```
        private void checkBoxDisplaySecondaryTile_Checked(object sender, RoutedEventArgs e)
        {
            // Look to see whether the Tile already exists and if so, don't try to create it
again.
            ShellTile TileToFind = ShellTile.ActiveTiles.FirstOrDefault(x =>
x.NavigationUri.ToString().Contains("DefaultTitle=FromTile"));

            // Create the Tile if we didn't find that it already exists.
            if (TileToFind == null)
            {
                // Create the Tile object and set some initial properties for the Tile.
                // The Count value of 12 shows the number 12 on the front of the Tile. Valid
values are 1-99.
                // A Count value of 0 indicates that the Count should not be displayed.
                StandardTileData NewTileData = new StandardTileData
                {
                    BackgroundImage = new Uri("Red.jpg", UriKind.Relative),
                    Title = "Secondary Tile",
                    Count = 12,
                    BackTitle = "Back of Tile",
                    BackContent = "Welcome to the back of the Tile",
                    BackBackgroundImage = new Uri("Blue.jpg", UriKind.Relative)
                };

                // Create the Tile and pin it to Start. This will cause a navigation to Start
and a deactivation of our application.
                ShellTile.Create(new Uri("/SecondaryTile.xaml?DefaultTitle=FromTile",
UriKind.Relative), NewTileData);
            }
        }
```

26. Add the following code to the **checkBoxDisplaySecondaryTile_Unchecked** event handler.

This code searches for the Tile with the ID that we gave it and then, if found, remove it from Start.

```
        // Event handler for when the check box is unchecked.  Delete the secondary Tile
        // if it exists.
        private void checkBoxDisplaySecondaryTile_Unchecked(object sender, RoutedEventArgs e)
        {
            // Find the Tile we want to delete.
            ShellTile TileToFind = ShellTile.ActiveTiles.FirstOrDefault(x =>
x.NavigationUri.ToString().Contains("DefaultTitle=FromTile"));

            // If the Tile was found, then delete it.
            if (TileToFind != null)
            {
                TileToFind.Delete();
            }

        }
```

27. Add each of the **button_Click** event handlers. Each event handler looks for the Tile to update, gets the new value from the corresponding text box, and updates the Tile with the new value. **[Copy and paste the code from Code.txt]**
28. You have now completed the code to create, delete, and update secondary Tiles.
29. To test your application:
    - Press F5 to start debugging your application.
    - Navigate to the **SecondaryTile** page and check the check box to create the secondary Tile. The phone will navigate to Start and the new Tile will appear.
    - Press the **Back** button on the phone emulator to navigate back to your application. By pressing the **Back** button, your debugging session will not end.
    - Enter a new value for the **Title** property and click the **Set Title** button.
    - Navigate back to **Start** by pressing the **Start** button on the phone emulator. The new **Title** property should be displayed.
    - Press the **Back** button again and try changing the other properties.
    - From Start, try tapping the secondary Tile. The **LiveTileLab** application will start and navigate directly to the **SecondaryTile** page. The default value of **Title** will be **FromTile**.