Uri Mapping in Windows Phone 7

This article explains how to use UriMapping in Windows Phone 7.

18 Dec 2011

Introduction

UriMapping is used to define a mapping (alias) between a short URI and the full path of the XAML pages within a project. Use of an alias URI empowers developers to change internal structure of a project by avoiding the following -

- Change of any navigation code.
- Breaking any external links to parts of the UI (for example from toast notifications).

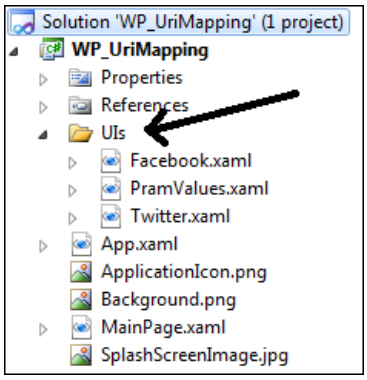
The mechanism also provides an easy way to pass data between pages as parameters, and to navigate to dynamically created pages that are named using the parameters.

This code example shows how to navigate between URI-mapped pages and how to pass parameters to a newly opened page.

Code example main view

URL mapping in code

In this code example we have created three XAML pages: Facebook.xaml, PramValues.xaml, Twitter.xaml, which are all stored in a single "UI" directory as shown below:



The following steps show how to create mappings between the URIs and these pages so that the pages are loaded when their respective buttons are clicked.

Step-1: Create an instance of UriMapper class object in **App.xaml.cs** class and add URI mapping items in Constructor for the Application object. Now add `UriMapper` object to PhoneApplicationFrame's object RootFrame's property UriMapper. See below code snippet for the same.

(C) Copyright Nokia 2014. All rights reserved.

Code Snippet:

```
public App()
 {
            // Global handler for uncaught exceptions.
            UnhandledException += Application_UnhandledException;
            // Standard Silverlight initialization
            InitializeComponent();
            // Phone-specific initialization
            InitializePhoneApplication();

            // URI Mapping
            var appNavigationMapper = new UriMapper();
            appNavigationMapper.UriMappings.Add(CreateUriMapping("Facebook",
"/UIs/Facebook.xaml"));
            appNavigationMapper.UriMappings.Add(CreateUriMapping("Twitter",
"/UIs/Twitter.xaml"));

appNavigationMapper.UriMappings.Add(CreateUriMapping("About/{value}/{value1}",
"/UIs/PramValues.xaml?id={value}&name={value1}"));

            RootFrame.UriMapper = appNavigationMapper;

            // Show graphics profiling information while debugging.
            if (System.Diagnostics.Debugger.IsAttached)
            {
                // Display the current frame rate counters.
                Application.Current.Host.Settings.EnableFrameRateCounter = true;

                // Show the areas of the app that are being redrawn in each frame.
                //Application.Current.Host.Settings.EnableRedrawRegions = true;

                // Enable non-production analysis visualization mode,
                // which shows areas of a page that are handed off to GPU with a colored
overlay.
                //Application.Current.Host.Settings.EnableCacheVisualization = true;

                // Disable the application idle detection by setting the
UserIdleDetectionMode property of the
                // application's PhoneApplicationService object to Disabled.
                // Caution:- Use this under debug mode only. Application that disables
user idle detection will continue to run
                // and consume battery power when the user is not using the phone.
                PhoneApplicationService.Current.UserIdleDetectionMode =
IdleDetectionMode.Disabled;
            }
  }
```

Step-2: Now we will design the UI (user interface) part to access the particular module using UriMapping mechanism. In this example I have added three buttons for Twitter, Facebook and Param(s) to call the specific UI with an action of appropriate command button. Here is the UI design with xaml code.

```
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
```


(C) Copyright Nokia 2014. All rights reserved.

```
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--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="MY APPLICATION"
Style="{StaticResource PhoneTextNormalStyle}"/>
            <TextBlock x:Name="PageTitle" Text="Uri Mapper" Margin="9,-7,0,0"
Style="{StaticResource PhoneTextTitle1Style}"/>
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <Button Content="Twitter" Height="72" HorizontalAlignment="Left"
Margin="40,68,0,0" Name="button1" VerticalAlignment="Top" Width="160"
Click="button1_Click" Background="#FFEB3636"></Button>
            <Button Content="Facebook" Height="72" HorizontalAlignment="Left"
Margin="240,209,0,0" Name="button2" VerticalAlignment="Top" Width="160"
Click="button2_Click" Background="#FFEB3636"></Button>
            <Button Content="Param(s)" Height="72" HorizontalAlignment="Left"
Margin="42,365,0,0" Name="button3" VerticalAlignment="Top" Width="160"
Click="button3_Click" Background="#FFEB3636" />
        </Grid>
    </Grid>
```

Step-3: At the end we will call the URI mapping using `NavigationService.Navigate()` method. In this example I have created a generic method for calling specific UI call by specific alias of UriMapping name as we mentioned in the **App.xaml.cs** class constructor. See below code snippet.

```
        private void button1_Click(object sender, RoutedEventArgs e)
        {
            DoNavigate("Twitter");
        }

        private void button2_Click(object sender, RoutedEventArgs e)
        {
            DoNavigate("Facebook");
        }

        private void button3_Click(object sender, RoutedEventArgs e)
        {
            DoNavigate("About/2/pavan"); // With two parameters
        }

        private void DoNavigate(string NavigatePageName)
        {
            this.NavigationService.Navigate(new Uri(NavigatePageName,
UriKind.Relative));
        }
```

In the above code, `DoNavigate("About/2/pavan")` is mapped as an address with two parameters, e.g. `About/{value}/{value1}` , to `{{{1}}}`. Note that the address used above is for example only, and does not map to a real page.

Step-4: Press F5 to execute the application.

(C) Copyright Nokia 2014. All rights reserved.

How to get query string values in navigated page

Sometimes you want to pass parameters to the page itself (rather than just selecting a page based on parameters). This can be done using NavigationContext to retrieve the specific values passed as parameters.

```
        void PramValues_Loaded(object sender, RoutedEventArgs e)
        {
            // Method 1: Get individual parameters
            textBlock1.Text = string.Format("Id= {0}",
NavigationContext.QueryString["id"]);
            textBlock2.Text = string.Format("Name= {0}",
NavigationContext.QueryString["name"]);

            // Method 2: Write all parameters to an array
            var values = NavigationContext.QueryString.ToArray();

            foreach (var item in values)
            {
                System.Diagnostics.Debug.WriteLine("Key : " + item.Key + "  " + "Value :
" + item.Value);
            }
        }
```

URI mapping using XAML

This section shows how you can create a URI mapping in XAML rather than in code.

Step-1: Go to **App.xaml** file and add the following namespace. xmlns:appnavigation="clr-namespace:System.Windows.Navigation;assembly=Microsoft.Phone"

Step-2: Add the XAML code in to the <Application.Resources> tag. In the xaml code you will see how to implement mapping with pass parameter between pages.

```
<Application.Resources>
    <appnavigation:UriMapper x:Key="UriMapper">
        <appnavigation:UriMapper.UriMappings>
            <appnavigation:UriMapping Uri="/Facebook" MappedUri="/Uis/Page1.xaml" />
            <appnavigation:UriMapping Uri="/Twitter/" MappedUri="/Uis/Twitter.xaml" />
            <appnavigation:UriMapping Uri="/About/{value1}/{value2}"
MappedUri="/Uis/About.xaml?id={id}&name={value1}" />
        </appnavigation:UriMapper.UriMappings>
    </appnavigation:UriMapper>
</Application.Resources>
```

If you want to link to a bunch of numbered pages without having to reference them individually (perhaps they are created dynamically) you can do this by specifying a parameter in the XAML URI definition. For example, to link to page1.xaml, page2.xaml and page3.xaml we can pass the parameter "{number}" as shown below:

```
<Application.Resources>
    <appnavigation:UriMapper x:Key="UriMapper">
        <appnavigation:UriMapper.UriMappings>
            <appnavigation:UriMapping Uri="/AppPage/{number}"
MappedUri="/Page{number}.xaml" />
        </appnavigation:UriMapper.UriMappings>
    </appnavigation:UriMapper>
</Application.Resources>
```


(C) Copyright Nokia 2014. All rights reserved.

Step-3: Finally add the Resources of `UriMapper` to Application `RootFrame.UriMapper` in code behind of **App.xaml** file inside the app method immediately after `InitializePhoneApplication()`.

```
RootFrame.UriMapper = Resources["UriMapper"] as UriMapper;
```

Download source code for this example:

File:WP UriMapping.zip

Summary

UriMapping is a very new mechanism in Windows Phone 7 (mango). It helps developer in re-structuring without modifying the code.

~- Happy Coding -~

(C) Copyright Nokia 2014. All rights reserved.