Listbox handling in Windows Phone

This article demonstrates how to implement a ListBox and get the value of the selected list item.

Introduction

`Listbox` can be implemented in two ways :

- Using `ListBoxItems`
- Data Binding

Generally `ListBox` is a kind of `ItemControl` together with `ListPicker` and other similar controls. The basic structure is same along with `SelectedItem` behavior. First we will use `ListBoxItems` method for creating the `ListBox`.

Using ListBoxItems

To make a list box using `ListBoxItems` method, the steps are listed below:

- Create a new project by going through **File | New Project | Windows Phone Application - Visual C#**. Give a desired project name.
- From Solution Explorer, open **MainPage.xaml** file. (If Solution Explorer window is currently not opened then open it via View>>Other Windows>>Solution Explorer)
- Inside Control Panel code fragment, add following code to create a list box with some items:

```
<!--ContentPanel - place additional content here-->
      <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
          <ListBox Margin="0,0,0,0" Name="lstCountries"
                   SelectionChanged="lstCountries_SelectionChanged">
              <ListBox.Items>
                  <ListBoxItem Content="Brazil"/>
                  <ListBoxItem Content="China"/>
                  <ListBoxItem Content="India"/>
                  <ListBoxItem Content="Switzerland"/>
                  <ListBoxItem Content="Austria"/>
              </ListBox.Items>
          </ListBox>
      </Grid>
  </Grid>
```

- We have given a value `lstCountries_SelectionChanged` to the property `SelectionChanged` of the `ListBox` control it means when ever the selection event will be fired on any list box item then the corresponding `lstCountries_SelectionChanged()` will be executed.
- In the **MainPage.xaml.cs** file, make a function like shown below:

```
 private void lstCountries_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            MessageBox.Show("List Item " + (lstCountries.SelectedIndex + 1) + "
pressed.");
        }
```

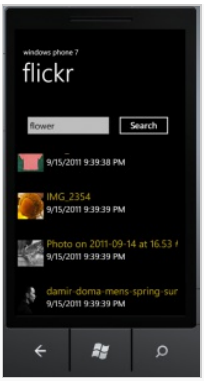
Using the above code, whenever a list item will be selected, a message box will prompt with the list item index being selected.

Using Data Binding Method

For this example we will use Flickr API to get the XML data and then display the content as List items. When user clicks on any of the list item, the detail page comes up.

Windows Phone	Windows Phone

(C) Copyright Nokia 2014. All rights reserved.



Flickr Search Page

Flickr Detail Page

Step by Step Implementation

- Download and install the Windows Phone SDK (WP SDK) 7.1 Release Candidate (RC)
- After you have downloaded and installed the Windows Phone SDK, launch the *Visual Studio 2010 Express for Windows Phone* and then choose *File->New Project*. Here, you will find the usual list of project templates, select *Silverlight for Windows Phone* in *Visual C#* section and then select *Windows Phone Application* template. As of now we will go with very basic template but you can implement the same project with other templates too.
- After creating a new project, you will get a view of the design surface and markup. Notice that the design surface shows the phone UI, letting you easily see how your application will look while you develop. For those familiar with Visual Studio, you will also find the familiar ToolBox, Solution Explorer and Properties pane.
- For our application, we will start by adding a `TextBox` and a `Button` from the Toolbox. We will change the name of the `TextBox` that we added to `SearchBox` and change the page title text to **flickr**.
- We will then write some code by double-clicking on the button and create an event handler in the code-behind file (**MainPage.xaml.cs**).

MainPage.xaml.cs

```
private void FlickrSearch_Click(object sender, RoutedEventArgs e)
        {
          WebClient webclient = new WebClient();
          webclient.DownloadStringCompleted += new
DownloadStringCompletedEventHandler(webclient_DownloadStringCompleted);
          webclient.DownloadStringAsync(new
Uri("http://api.flickr.com/services/feeds/photos_public.gne?tag=" + SearchBox.Text +
"&format=rss2")); // Flickr search
      }
```

We took the advantage of WebClient networking class to asynchronously download a Flickr feed. This takes three lines of code in total: (1) declaring and creating the `WebClient`, (2) attaching an event handler and then (3) calling the asynchronous `DownloadStringAsync` method.In the `DownloadStringAsync` call, we will pass a Flickr `Uri` plus a query string which pulls the text from the `SearchBox` TextBox. This feed will pull all the respective search item in an XML format. When the call completes, the `DownloadStringCompleted` event is fired and our generated event handler `webclient_DownloadStringCompleted` will be called

```
void webclient_DownloadStringCompleted(object sender, DownloadStringCompletedEventArgs
e)
   {
           if (e.Error != null)
            {
               MessageBox.Show("error");
            }
      // parsing Flickr
             XElement XmlTweet = XElement.Parse(e.Result);
           //var ns = XmlTweet.GetDefaultNamespace(); // get the default namespace
             XNamespace ns = "http://search.yahoo.com/mrss/";
            listBox1.ItemsSource = from tweet in XmlTweet.Descendants("item")
            select new FlickrData
             {
               ImageSource = tweet.Element(ns + "thumbnail").Attribute("url").Value,
               Message = tweet.Element("description").Value,
               UserName = tweet.Element("title").Value,
               PubDate = DateTime.Parse(tweet.Element("pubDate").Value)
             };
      }
```


(C) Copyright Nokia 2014. All rights reserved.

The result returned from the Flickr call will be in XML format. To parse this we'll use LINQ to XML. LINQ to XML lets us create simple queries for accessing data in an xml feed. To use this library, we will first need to add a reference to the assembly (right click on the References folder in the solution explorer and choose *Add Reference*), and then select **System.Xml.Linq** from the list. Then add a *using System.Xml.Linq* namespace reference at the top of the code-behind file at the top of **Main.xaml.cs** file. `XElement` is used to parse the XML data. To store the data we will create a helper class `FlickrData`

```
  public class FlickrData
         {
           public string UserName { get; set; }
           public string Message { get; set; }
           public string ImageSource { get; set;}
           public DateTime PubDate { get; set; }
         }
```

Then with the use of LINQ to XML we parse the returned XML string from Flickr. What the query does is pull out the key pieces of information for each Flickr post from the `SearchBox` we passed as the query string. For each item in the XML, we are creating a new `FlickrData` in the `IEnumerable<XElement>` returned by the Linq query. We then assign the generated `FlickrData` sequence to the ListBox's `ItemsSource` property. We will then follow one more step to complete this step. In the **MainPage.xaml** file, we will add an `ItemTemplate` to the `ListBox`. For the demo, I have used a simple template that uses databinding to show the content.

```
<ListBox SelectionChanged="listBoxFlickerSearch_SelectionChanged"  Height="496"
HorizontalAlignment="Left" Margin="-4,155,0,0" Name="listBox1" VerticalAlignment="Top"
Width="460" >
   <ListBox.ItemTemplate>
        <DataTemplate>
             <StackPanel Orientation="Horizontal" Height="132"
MouseLeftButtonDown="Tap_LeftButtonDown"
                  MouseLeftButtonUp="Tap_LeftButtonUp"
                   MouseLeave="Tap_MouseLeave">
                   <Image Source="{Binding ImageSource}" Height="73" Width="73"
VerticalAlignment="Top" Margin="0,10,8,0"/>
                    <StackPanel Width="370">
                          <TextBlock Text="{Binding UserName}" Foreground="#FFC8AB14"
FontSize="28" />
                          <!--TextBlock Text="{Binding Message}" TextWrapping="Wrap"
FontSize="24" /-->
                          <TextBlock Text="{Binding PubDate}" TextWrapping="Wrap"
FontSize="24" />
                     </StackPanel>
                  </StackPanel>
            </DataTemplate>
        </ListBox.ItemTemplate>
   </ListBox>
```

Now if you compile and run the application, it will come up the the list of Flickr data. but when you click on any of the item nothing happens.

- We then create another page **FlickrDetail.xaml** to display the selected item. Right click on the *Project->Add->New Item->Window Phone Portrait*. In the **FlickrDetail.xaml** file, we then add an `ItemTemplate` to the `ListBox` as we did before, this list box we contain the detail of the item.

FlickrDetail.xaml

```
<ListBox Height="541" HorizontalAlignment="Left" Margin="9,35,0,0"
Name="listBoxFlickrDetail" VerticalAlignment="Top" Width="441" >
<ListBox.ItemTemplate>
<DataTemplate>
   <StackPanel Orientation="Horizontal" Height="132">
       <Image Source="{Binding ImageSource}" Height="73" Width="73"
VerticalAlignment="Top" Margin="0,10,8,0"/>
             <StackPanel Width="370">
                     <TextBlock Text="{Binding UserName}" Foreground="#FFC8AB14"
FontSize="28" />
                     <TextBlock Text="{Binding PubDate}" TextWrapping="Wrap"
FontSize="24" />
                    <TextBlock Text="{Binding Message}" TextWrapping="Wrap"
FontSize="24" />
             </StackPanel>
        </StackPanel>
```


(C) Copyright Nokia 2014. All rights reserved.

```
    </DataTemplate>
 </ListBox.ItemTemplate>
 </ListBox>
```

- In **MainPage.xaml** we add `SelectionChanged+=listBoxFlickerSearch_SelectionChanged` to the `ListBox` control to get the selected item. In **MainPage.xaml.cs** we fetch the selected item and navigate the screen with `NavigationService`

```
private void listBoxFlickerSearch_SelectionChanged(object sender,
SelectionChangedEventArgs e)
 {
if (e.AddedItems.Count > 0)
  {
// get RSS item from ListBox and store in class var. Store page to navigate to in class
var.
selectedItem = (FlickrData)e.AddedItems[0];
targetPage = new Uri("/FlickrDetail.xaml", UriKind.Relative);
// reset selection of ListBox
((ListBox)sender).SelectedIndex = -1;
                // change page navigation
 NavigationService.Navigate(targetPage);
FrameworkElement root = Application.Current.RootVisual as FrameworkElement;
 root.DataContext = selectedItem;
}
}
```

- In *FlickrDetail.xaml* we create a class to hold the selected item data.

```
public class FlickrDetailPage
{
public string UserName { get; set; }
public string Message { get; set; }
public string ImageSource { get; set; }
public DateTime PubDate { get; set; }
}
```

and assign the selected item value to the accessor while loading the page

```
void RssItemPage_Loaded(object sender, RoutedEventArgs e)
{                       // Set page title to news title and navigate to string
containing news item.          //  PageTitle.Text =
((RssItem)DataContext).Title.ToLower();
 FlickrDetailPage iFlickrDetail = new FlickrDetailPage();
iFlickrDetail.UserName = ((FlickrData )DataContext).UserName.ToLower();
iFlickrDetail.ImageSource = ((FlickrData)DataContext).ImageSource.ToLower();
iFlickrDetail.PubDate = ((FlickrData)DataContext).PubDate;
iFlickrDetail.Message = ((FlickrData)DataContext).Message.ToLower();
listBoxFlickrDetail.Items.Add(iFlickrDetail);

}
```

Finally we then assign the generated `FlickrDetailPage` sequence to the List Box's `ItemsSource` property.

- Compile the application with *Ctrl+Shift+B* and then run it with **Crtl+F5** in release mode.

Source Code

The full source code of the example is available here: File:FlickrWP7.zip

(C) Copyright Nokia 2014. All rights reserved.

(C) Copyright Nokia 2014. All rights reserved.