Picasa Image Gallery with JSON in WP7

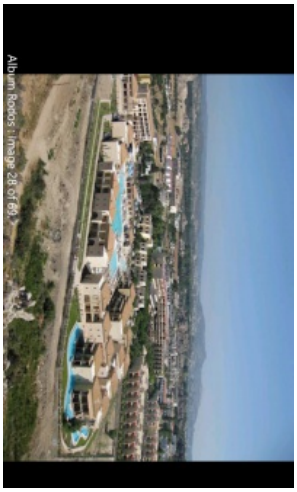
This article explains how to create a Picasa Image Gallery with JSON in Windows Phone 7. We create an application called MyPicasa to illustrate this.

06 May 2012

Introduction

This article is intends to show how one can access image albums (private/public/shared) in Google Picasa. Though there are several ways offered from Google to get image album data, "JSON" is covered in this article for fetching data.

MyPicasa application first authenticates to Google and get image albums data with JSON string (image 1). All the image album covers and title texts are displayed in vertically scrollable ListBox (image 2). Selected album's thumbnail images are loaded and displayed when user selects album from ListBox (image 3). Selected album images can be shown in portrait or landscape mode. User can show next or previous image using with Gestures (image 4).



Here is a small demo video, which shows how the application works: The media player is loading...

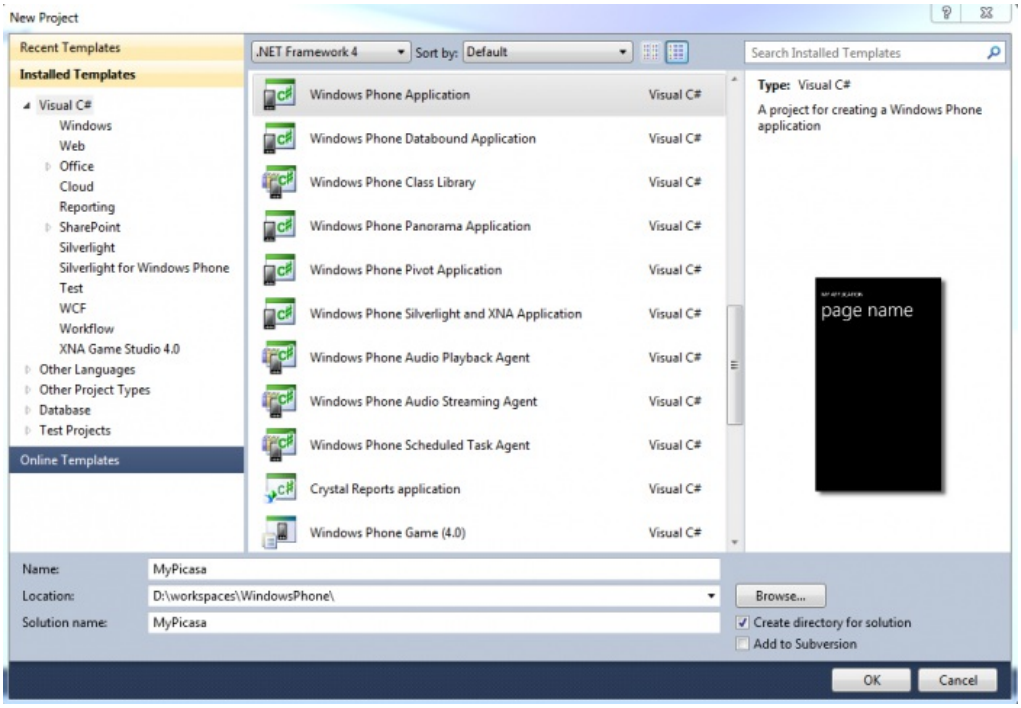
Windows Phone 7.1 SDK

To Develop application for Windows Phone 7 devices, you need to install Windows Phone 7.1 SDK. You can download latest SDK for Windows Phone here.

Windows Phone Application

To start creating a new Windows Phone application, start Microsoft Visual Studio then create a new Project and select Windows Phone Application Template.

(C) Copyright Nokia 2013. All rights reserved.



In this example we have used C# as code behind language.

Project extensions

Microsoft Visual Studio extensions can be installed in many different ways. In this article we will use Nuget which makes extension installation easier. In this article we will need Silverlight Toolkit and Simple JSON extensions.

Nuget

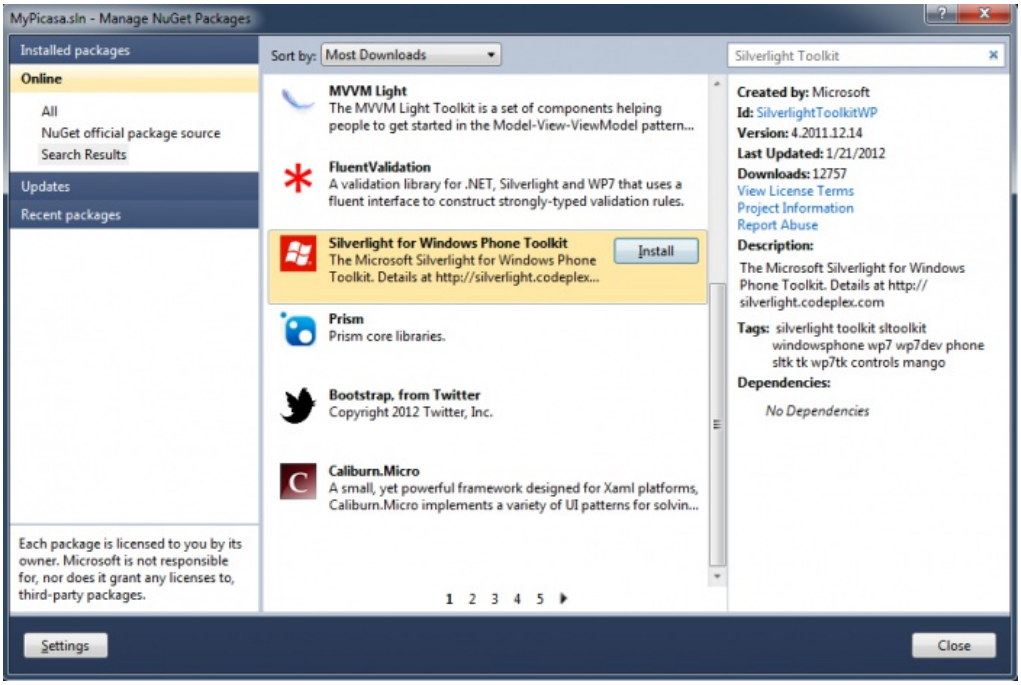
"NuGet is a Visual Studio extension that makes it easy to add, remove, and update libraries and tools in Visual Studio projects that use the .NET Framework. When you add a library or tool, NuGet copies files to your solution and automatically makes whatever changes are needed in your project, such as adding references and changing your app.config or web.config file. When you remove a library, NuGet removes files and reverses whatever changes it made in your project so that no clutter is left."

More information about Nuget can be found here (and installation package).

Silverlight Toolkit

Silverlight Toolkit has many great extensions to Windows Phone developers. In this article we will use WrapPanel Control to layout Album thumbnails to the screen. You can add Silverlight Toolkit to your project with Nuget in Visual Studio:

Select: Tools > Library Package Manager... > Manage Nuget Packages for Solutions... and search Silverlight Toolkit



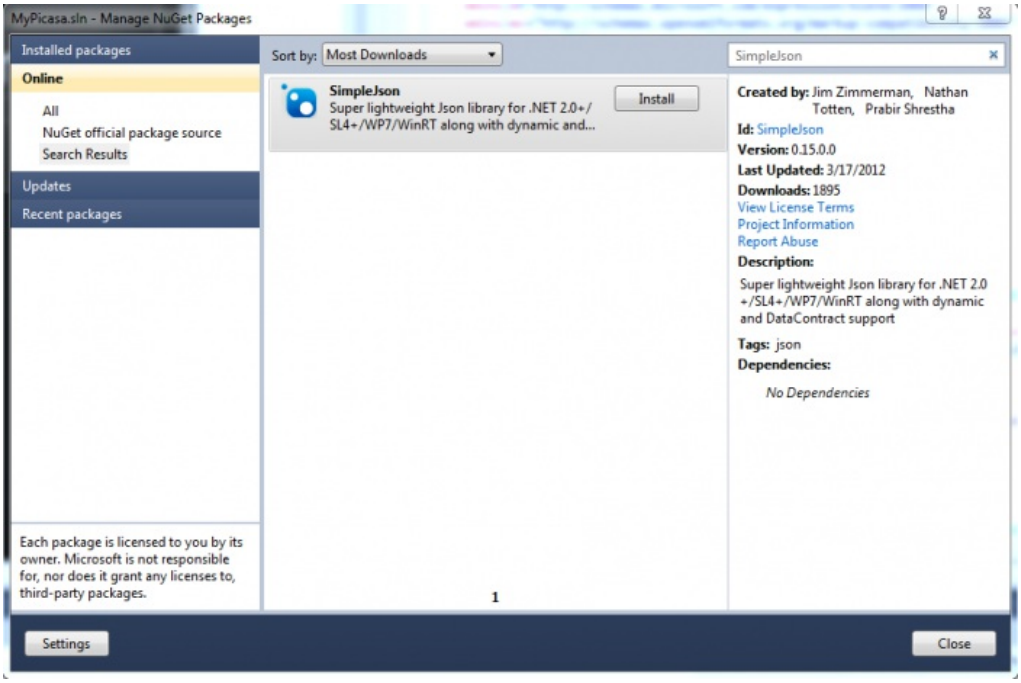
(C) Copyright Nokia 2013. All rights reserved.

More information to Silverlight Toolkit can be found here.

Simple JSON

Here in Nokia Wiki there are quite a lot of coding examples covered with XML data in WP7. This time we want to make an example with JSON. One possible extension is to use Simple JSON to handle JSON data from Google. You can add Simple JSON to your project with Nuget:

Select: Tools > Library Package Manager... > Manage Nuget Packages for Solutions... and search SimpleJSON



More information to Simple JSON can be found here.

Loading and displaying many images from net

There can be a small issue when you try to load lot of images from the web at the same time in Windows Phone. You can find a few assemblies from the net to fix this. One good one is PhonePerformance assembly (you can download compiled PhonePerformance assembly with source codes here). You have to download and unzip it. Copy PhonePerformance.dll to your project and add a new reference to it with Solution Explorer.

You might get warning asking you to unblock PhonePerformance.dll assembly (because it is downloaded from web). Close your Visual Studio. Open Windows Explorer and browse your PhonePerformance.dll file. Open file properties and Unblock.

Collection classes used in this code example

There are a lot of different kind of data available in Picasa, only the following data is used with albums and images in this code example.

Album.cs

Album class is used to store album information: title, published time, href to album and cover thumbnail.

To create New Classes in your project:

1. Right click your project in Solutions Explorer
2. Select Add and then Class...

```
namespace MyPicasa
{
    public class Album
    {
        public string title { get; set; }
        public string published { get; set; }
        public string href { get; set; }
        public string thumbnail { get; set; }
    }
}
```

AlbumImage.cs

AlbumImage class is used to store image information like image title, content url, size and thumbnail url.

```
namespace MyPicasa
{
```


(C) Copyright Nokia 2013. All rights reserved.

```
    public class AlbumImage
    {
        public string title { get; set; }
        public string content { get; set; }
        public string width { get; set; }
        public string height { get; set; }
        public string size { get; set; }
        public string thumbnail { get; set; }
    }
}
```

AlbumImages.cs

AlbumImages Class is a collection of album images.

```
namespace MyPicasa
{
    public class AlbumImages : ObservableCollection<AlbumImage> { }
}
```

AlbumList.cs

AlbumList Class is a collection of albums.

```
namespace MyPicasa
{
    public class AlbumList : ObservableCollection<Album> { }
}
```

Login Page

User can give login information (email and password) in Login Page. Google authentication is checked with these information. If the optional username TextBlock is empty, then user's own public and private albums are loaded. User can add his/her friend username here to get shared albums downloaded instead.

Design (MainPage.xaml)

Login View is a simple grid with basic controls.

```
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
    <TextBlock Height="30" HorizontalAlignment="Left" Margin="14,29,0,0"
        Text="GOOGLE ACCOUNT" VerticalAlignment="Top" />
    <TextBlock Height="30" HorizontalAlignment="Left" Margin="10,80,0,0"
        Text="Email:" VerticalAlignment="Top" />
    <TextBox Height="72" HorizontalAlignment="Left" Margin="5,110,0,0"
        x:Name="EmailTextBox" Text="" VerticalAlignment="Top" Width="460" />
    <TextBlock Height="30" HorizontalAlignment="Left" Margin="10,190,0,0"
        Text="Password:" VerticalAlignment="Top" />
    <PasswordBox Height="72" HorizontalAlignment="Left" Margin="5,220,0,0"
        x:Name="PasswordTextBox" VerticalAlignment="Top" Width="460" />
    <TextBlock Height="30" HorizontalAlignment="Left" Margin="10,300,0,0"
        Name="UsernameTextBlock" Text="Username: (optional, email without @google.com)" VerticalAlignment="Top" />
    <TextBox Height="72" HorizontalAlignment="Left" Margin="5,330,0,0"
        x:Name="UsernameTextBox" Text="" VerticalAlignment="Top" Width="460" />
    <CheckBox Content="Save password" Height="72" HorizontalAlignment="Left" Margin="10,415,0,0"
        x:Name="SavePasswordCheckBox" VerticalAlignment="Top" />
    <Button Content="Login" Height="72" HorizontalAlignment="Left" Margin="149,504,0,0"
        x:Name="LoginButton" VerticalAlignment="Top" Width="160" Click="LoginButton_Click" />
</Grid>
```

Programming (MainPage.xaml.cs)

User's email and password are stored to Isolated Storage, so user doesn't need to type those when application is starting again.

(C) Copyright Nokia 2013. All rights reserved.

Main Page uses a few member variables to store user information.

```
private IsolatedStorageSettings appSettings;
private const string emailKey = "emailKey";
private const string passwordKey = "passwordKey";
private const string savepasswordKey = "savepasswordKey";
private const string usernameKey = "usernameKey";

private string username = "";
private string email = "";
private string password = "";
private bool savepassword = false;

public MainPage()
{
    InitializeComponent();
    appSettings = IsolatedStorageSettings.ApplicationSettings;
}
```

`OnNavigatedTo()`-method will be called when this page is displayed. Here we first load stored email and password from Isolaged Storage and show those in TextBox and CheckBox controls.

```
protected override void OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e)
{
 // load and show saved email from isolated storage
 if (appSettings.Contains(emailKey))
 {
  email = (string)appSettings[emailKey];
 }
 EmailTextBox.Text = email;

 // load password from isolated storage
 if (appSettings.Contains(passwordKey))
 {
  password = (string)appSettings[passwordKey];
 }

 // username from isolated storage
 if (appSettings.Contains(usernameKey))
 {
  username = (string)appSettings[usernameKey];
 }
 UsernameTextBox.Text = username;

 // show password if selected to save
 if (appSettings.Contains(savepasswordKey))
 {
  string savepass = (string)appSettings[savepasswordKey];
  if (savepass == "true")
  {
   savepassword = true;
   PasswordTextBox.Password = password;
  }
  else
  {
   savepassword = false;
   PasswordTextBox.Password = "";
  }
  SavePasswordCheckBox.IsChecked = savepassword;
 }
}
```

Albums Page is loaded when Login Button is pressed here in the Login Page.

```
private void LoginButton_Click(object sender, RoutedEventArgs e)
{
 username = UsernameTextBox.Text;
 email = EmailTextBox.Text;
 password = PasswordTextBox.Password;
 savepassword = (bool)SavePasswordCheckBox.IsChecked;
 this.NavigationService.Navigate(new Uri("/AlbumsPage.xaml", UriKind.Relative));
}
```

`OnNavigatedFrom()`-method will be called when user is leaving from this page. Here we save user information to Isolaged Storage.

```
protected override void OnNavigatedFrom(System.Windows.Navigation.NavigationEventArgs e)
{
 // add email to isolated storage
 appSettings.Remove(emailKey);
 appSettings.Add(emailKey, email);

 // add savepassword and password to isolated storage
 appSettings.Remove(savepasswordKey);
 appSettings.Remove(passwordKey);

 // add username to isolated storage
 appSettings.Remove(usernameKey);
 appSettings.Add(usernameKey, username);

 if (SavePasswordCheckBox.IsChecked==true)
 {
  appSettings.Add(savepasswordKey, "true");
  appSettings.Add(passwordKey, password);
 }
 else
 {
  appSettings.Add(savepasswordKey, "false");
  appSettings.Add(passwordKey, password);
 }
}
```

Albums Page

Albums Page display user's Albums in WrapPanel Control.

(C) Copyright Nokia 2013. All rights reserved.

Design (AlbumsPage.xaml)

Page grid contains `ListBox` which `ItemsPanel` has a `WrapPanel` Control. This is nice and easy way to get content layout horizontally and vertically as a tile view. Silverlight Toolkit offer also `TiltEffect` which is enabled in every album. So every time when user clicks album cover it moves down a little. Album cover and title are stored in `StackPanel` which uses vertical orientation.

```
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
 <ListBox x:Name="AlbumsListBox"
    toolkit:TiltEffect.IsTiltEnabled="True"
    SelectionChanged="AlbumsListBox_SelectionChanged">
  <ListBox.ItemsPanel>
   <ItemsPanelTemplate>
    <toolkit:WrapPanel
     HorizontalAlignment="Left"
     Margin="0,0,0,0"
     VerticalAlignment="Top"
      />
   </ItemsPanelTemplate>
  </ListBox.ItemsPanel>
  <ListBox.ItemTemplate>
   <DataTemplate>
    <StackPanel Margin="5">
     <StackPanel Orientation="Vertical">
      <Image delay:LowProfileImageLoader.UriSource="{Binding thumbnail}" Width="160" Height="160"/>
      <TextBlock Text="{Binding title}"  Width="180" Height="30"/>
     </StackPanel>
    </StackPanel>
   </DataTemplate>
  </ListBox.ItemTemplate>
 </ListBox>
</Grid>
```

Programming (AlbumsPage.xaml.cs)

`OnNavigatedTo()`-method will be called when this page is displayed. Here we first load stored email, password and optional username from Isolated Storage and define data feed URL. If optional username is defined in Login Page, then this username is included to data feed URL. After that we will start to get authentication from Google.

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

 // load saved username from isolated storage
 if (appSettings.Contains(usernameKey))
 {
  string differentUsername = (string)appSettings[usernameKey];
  if (differentUsername != "") username = differentUsername;
 }

 // load saved email from isolated storage
 if (appSettings.Contains(emailKey))
 {
  email = (string)appSettings[emailKey]; // for example firstname.lastname@gmail.com
  if (username == "" && email.IndexOf("@") != -1) username = email.Substring(0, email.IndexOf("@")); // firstname.lastname
  dataFeed = String.Format("http://picasaweb.google.com/data/feed/api/user/{0}?alt=json", username);
 }

 // load password from isolated storage
 if (appSettings.Contains(passwordKey))
 {
  password = (string)appSettings[passwordKey];
 }

 // we are coming back from AlbumPage
 if (e.NavigationMode == System.Windows.Navigation.NavigationMode.Back)
 {
  AlbumsListBox.ItemsSource = app.albums;
  AlbumsListBox.SelectedIndex = -1;
 }
 else
 {
  // get authentication from Google
  GetAuth();
 }
}
```


(C) Copyright Nokia 2013. All rights reserved.

To get authentication to use Picasa services, we have to use `WebClient` class to download authentication information from Google. With Picasa we have to use "lh2" as a service and accountType is "GOOGLE". You can find more information about Google services with Picasa here.

```
private void GetAuth()
{
 string service = "lh2"; // Picasa
 string accountType = "GOOGLE";

 WebClient webClient = new WebClient();
 Uri uri = new Uri(string.Format("https://www.google.com/accounts/ClientLogin?Email={0}&Passwd={1}&service={2}&accountType={3}", ema
 webClient.DownloadStringCompleted += new DownloadStringCompletedEventHandler(AuthDownloaded);
 webClient.DownloadStringAsync(uri);
}
```

Google sends authentication code if login was successful. We can parse it from result and use it to load albums data from Google. Note: Authentication code is stored to `App.xaml.cs` file (and album and images data). We will discuss more about this later in this chapter (handling Tombstone mode).

```
private void AuthDownloaded(object sender, DownloadStringCompletedEventArgs e)
{
 try
 {
  if (e.Result != null && e.Error == null)
  {
   app.auth = "";
   int index = e.Result.IndexOf("Auth=");
   if (index != -1)
   {
    app.auth = e.Result.Substring(index + 5);
   }
   if (app.auth != "")
   {
    // get albums from Google
    GetAlbums();
    return;
   }
  }
  MessageBox.Show("Cannot get authentication from google. Check your login and password.");
 }
 catch (WebException)
 {
  MessageBox.Show("Cannot get authentication from google. Check your login and password.");
 }
}
```

`WebClient` class is used again to fetch albums data from Google. Note that we have to add authentication to our headers in `WebClient` object.

```
private void GetAlbums()
{
 WebClient webClient = new WebClient();
 webClient.Headers[HttpRequestHeader.Authorization] = "GoogleLogin auth=" + app.auth;
 Uri uri = new Uri(dataFeed, UriKind.Absolute);
 webClient.DownloadStringCompleted += new DownloadStringCompletedEventHandler(AlbumsDownloaded);
 webClient.DownloadStringAsync(uri);
}
```

There can be a lot of data returned (depends how much albums you have) from the Google. In this article all the data is stored to `App.xaml.cs` class to handle Tombstone mode in different views. We have covered Tombstone mode for example here in Movies in Finnkino Theatres article.

Now we only need to know that these variables are handled in `App.xaml.cs` and they are used in different Pages in this application:

```
public AlbumList albums = new AlbumList();
public AlbumImages albumImages = new AlbumImages();
public int selectedImageIndex;
public int selectedAlbumIndex;
public string auth;
```

Back to Albums data. If there are no errors, we will parse JSON string to our collections (`AlbumList`). First we have to deserialize JSON string to dynamic object, so we can start to find data from JSON string. If you look carefully what kind of JSON string Google returns, you can find for example following object: `feed`, `author` and `album entries` with many different details.

Try this link to see example sample JSON string from Picasa. In this article we will cover only few of those details.

Above `WebClient` class object will call `AlbumsDownloaded()`-method when data is returned from Google.

```
public void AlbumsDownloaded(object sender, DownloadStringCompletedEventArgs e)
{
 try
 {
  if (e.Result == null || e.Error != null)
  {
   MessageBox.Show("Cannot get albums data from Picasa server!");
   return;
  }
  else
  {
   // Deserialize JSON string to dynamic object
   IDictionary<string, object> json = (IDictionary<string, object>)SimpleJson.DeserializeObject(e.Result);
   // Feed object
   IDictionary<string, object> feed = (IDictionary<string, object>)json["feed"];
   // Author List
   IList author = (IList)feed["author"];
   // First author (and only)
   IDictionary<string, object> firstAuthor = (IDictionary<string, object>)author[0];
   // Album entries
   IList entries = (IList)feed["entry"];
   // Delete previous albums from albums list
   app.albums.Clear();
   // Find album details
   for (int i = 0; i < entries.Count; i++)
```


(C) Copyright Nokia 2013. All rights reserved.

```
    {
     // Create a new Album
     Album album = new Album();
     // Album entry object
     IDictionary<string, object> entry = (IDictionary<string, object>)entries[i];
     // Published object
     IDictionary<string, object> published = (IDictionary<string, object>)entry["published"];
     // Get published date
     album.published = (string)published["$t"];
     // Title object
     IDictionary<string, object> title = (IDictionary<string, object>)entry["title"];
     // Album title
     album.title = (string)title["$t"];
     // Link List
     IList link = (IList)entry["link"];
     // First link is album data link object
     IDictionary<string, object> href = (IDictionary<string, object>)link[0];
     // Get album data addres
     album.href = (string)href["href"];
     // Media group object
     IDictionary<string, object> mediagroup = (IDictionary<string, object>)entry["media$group"];
     // Media thumbnail object list
     IList mediathumbnailList = (IList)mediagroup["media$thumbnail"];
     // First thumbnail object (smallest)
     var mediathumbnail = (IDictionary<string, object>)mediathumbnailList[0];
     // Get thumbnail url
     album.thumbnail = (string)mediathumbnail["url"];
     // Add album to albums
     app.albums.Add(album);
    }
    // Add albums to AlbumListBox
    AlbumsListBox.ItemsSource = app.albums;
   }
  }
  catch (WebException)
  {
   MessageBox.Show("Cannot get albums data from Picasa server!");
  }
  catch (KeyNotFoundException)
  {
   MessageBox.Show("Cannot load images from Picasa Server - JSON parsing error happened!");
  }
 }
```

JSON string is deserialized to IDictionary object. This way we can search data from JSON string like a name value pairs. Above code has a lot of comments to help understand data parsing.

Now all the albums data from Picasa should be visible in Albums Page. Selected album images will be loaded and showed when user select album from AlbumListBox.

```
private void AlbumsListBox_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
 // If real selection is happened, go to a AlbumPage
 if (AlbumsListBox.SelectedIndex == -1) return;
 app.selectedAlbumIndex = AlbumsListBox.SelectedIndex;
 this.NavigationService.Navigate(new Uri("/AlbumPage.xaml?SelectedIndex=" + AlbumsListBox.SelectedIndex, UriKind.Relative));
}
```

Album Page

Selected album's thumbnail images will be visible in Album Page.

Design (AlbumPage.xaml)

Here we will use the same kind of technique which was used in the previous Albums Page. Grid contains `ListBox` which uses `WrapPanel` and finally one item in `WrapPanel` is only a `Image` Control. In this article we will use one of the smallest thumbnail image which is available from Google (72x54px). There can be a lot of thumbnails loading, and we are using here PhonePerformance assembly to help loading images (remember add reference to PhonePerformance.dll in your project).

```
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
 <ListBox x:Name="AlbumImagesListBox"
    toolkit:TiltEffect.IsTiltEnabled="True"
    SelectionChanged="AlbumImagesListBox_SelectionChanged">
  <ListBox.ItemsPanel>
```


(C) Copyright Nokia 2013. All rights reserved.

```
    <ItemsPanelTemplate>
     <toolkit:WrapPanel
      HorizontalAlignment="Left"
      Margin="0,0,0,0"
      VerticalAlignment="Top"
       />
    </ItemsPanelTemplate>
   </ListBox.ItemsPanel>
   <ListBox.ItemTemplate>
    <DataTemplate>
     <StackPanel Margin="5">
      <StackPanel Orientation="Vertical">
       <Image delay:LowProfileImageLoader.UriSource="{Binding thumbnail}" Width="72" Height="54"/>
      </StackPanel>
     </StackPanel>
    </DataTemplate>
   </ListBox.ItemTemplate>
  </ListBox>
 </Grid>
```

Programming (AlbumPage.xaml.cs)

`OnNavigatedTo()`-method will be called when this page is displayed. Here we first check that is user coming back from the Image Page (we don't have to load album images data again) or is user coming from Main Page (we have to start loading images data from Google).

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

 // We are coming back from Images Page
 if (e.NavigationMode == System.Windows.Navigation.NavigationMode.Back)
 {
  AlbumImagesListBox.ItemsSource = app.albumImages;
  PageTitle.Text = app.albums[app.selectedAlbumIndex].title;
  AlbumImagesListBox.SelectedIndex = -1;
  return;
 }

 // We are coming from MainPage, start loading album images
 IDictionary<string, string> parameters = this.NavigationContext.QueryString;
 if (parameters.ContainsKey("SelectedIndex"))
 {
  int selectedIndex = Int32.Parse(parameters["SelectedIndex"]);
  PageTitle.Text = app.albums[selectedIndex].title;
  GetImages(selectedIndex);
 }
}
```

If we are coming from Main Page, then we have to load images data from Google. We have to use `WebClient` and remember to use Authentication in headers.

```
private void GetImages(int selectedIndex)
{
 WebClient webClient = new WebClient();
 string auth = "GoogleLogin auth=" + app.auth;
 webClient.Headers[HttpRequestHeader.Authorization] = auth;
 Uri uri = new Uri(app.albums[selectedIndex].href, UriKind.Absolute);
 webClient.DownloadStringCompleted += new DownloadStringCompletedEventHandler(ImagesDownloaded);
 webClient.DownloadStringAsync(uri);
}
```

`ImagesDownloaded()`-method will be called when data is loaded from Google. If there are no errors, we will parse JSON data to our collections (`AlbumImages`). First we have to deserialize JSON String to object to get access to images data. JSON data contains for example `Feed` and `Image entries`. Data will be parsed a same way as we did in prevous page (now we are handing images data).

```
public void ImagesDownloaded(object sender, DownloadStringCompletedEventArgs e)
{
 try
 {
  if (e.Result == null || e.Error != null)
  {
   MessageBox.Show("Cannot load images from Picasa server!");
   return;
  }
  else
  {
   // Deserialize JSON string to dynamic object
   IDictionary<string, object> json = (IDictionary<string, object>)SimpleJson.DeserializeObject(e.Result);
   // Feed object
   IDictionary<string, object> feed = (IDictionary<string, object>)json["feed"];
   // Number of photos object
   IDictionary<string, object> numberOfPhotos = (IDictionary<string, object>)feed["gphoto$numphotos"];
   // Entries List
   var entries = (IList)feed["entry"];
   // clear previous images from albumImages
   app.albumImages.Clear();
   // Find image details from entries
   for (int i = 0; i < entries.Count; i++)
   {
    // Create a new albumImage
    AlbumImage albumImage = new AlbumImage();
    // Image entry object
    IDictionary<string, object> entry = (IDictionary<string, object>)entries[i];
    // Image title object
    IDictionary<string, object> title = (IDictionary<string, object>)entry["title"];
    // Get album title
    albumImage.title = (string)title["$t"];
    // Album content object
    IDictionary<string, object> content = (IDictionary<string, object>)entry["content"];
    // Get image src url
    albumImage.content = (string)content["src"];
    // Image width object
    IDictionary<string, object> width = (IDictionary<string, object>)entry["gphoto$width"];
    // Get image width
    albumImage.width = (string)width["$t"];
    // Image height object
```


(C) Copyright Nokia 2013. All rights reserved.

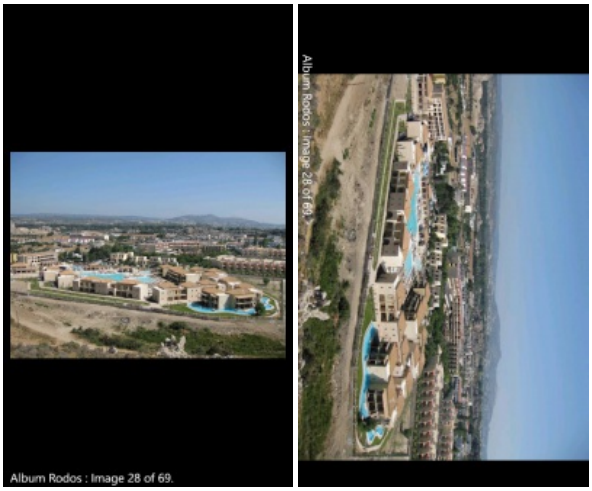
```
    IDictionary<string, object> height = (IDictionary<string, object>)entry["gphoto$height"];
    // Get image height
    albumImage.height = (string)height["$t"];
    // Image size object
    IDictionary<string, object> size = (IDictionary<string, object>)entry["gphoto$size"];
    // Get image size
    albumImage.size = (string)size["$t"];
    // Image media group List
    IDictionary<string, object> mediaGroup = (IDictionary<string, object>)entry["media$group"];
    IList mediaThumbnailList = (IList)mediaGroup["media$thumbnail"];
    // First thumbnail object
    IDictionary<string, object> mediathumbnail = (IDictionary<string, object>)mediaThumbnailList[0];
    // Get thumnail url
    albumImage.thumbnail = (string)mediathumbnail["url"];
    // Add albumImage to albumImages Collection
    app.albumImages.Add(albumImage);
   }
   // Add albumImages to AlbumImagesListBox
   AlbumImagesListBox.ItemsSource = app.albumImages;
  }
 }
 catch (WebException)
 {
  MessageBox.Show("Cannot load images from Picasa server!");
 }
 catch (KeyNotFoundException)
 {
  MessageBox.Show("Cannot load images from Picasa Server - JSON parsing error happened!");
 }
}
```

Now all the selected album images should be visible in Album Page. Selected image will be loaded and showed when the user select image from `AlbumImageListBox`.

```
private void AlbumImagesListBox_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
 // If real selection is happened, go to a ImagesPage
 if (AlbumImagesListBox.SelectedIndex == -1) return;
 this.NavigationService.Navigate(new Uri("/ImagesPage.xaml?SelectedIndex=" + AlbumImagesListBox.SelectedIndex + "&SelectedAlbum=" +
}
```

Images Page

Images Page shows selected album image. This page supports portrait and landscape orientations. User can swipe screen to show previous or next image in this album.



Design (ImagesPage.xaml)

When you create a new page you can select Portrait Page as a template. Later you can modify your page to support both landscape and portrait modes in Page `PhoneApplicationPage` root element:

```
<phone:PhoneApplicationPage
  ...
  SupportedOrientations="PortraitOrLandscape" Orientation="Landscape"
>
```

Page grid contains only `Image` and `TextBlock`, which is positioned at the bottom of the screen.

```
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
 <Image x:Name="image" HorizontalAlignment="Center" VerticalAlignment="Center"/>
 <TextBlock Height="30"
      Margin="0,0,0,0"
      HorizontalAlignment="Left"
      VerticalAlignment="Bottom"
      x:Name="ImageInfoTextBlock"
      Text=""
      Width="692" />
</Grid>
```


(C) Copyright Nokia 2013. All rights reserved.

Programming (ImagesPage.xaml)

`OnNavigatedTo()`-method will be called when this page is shown. Selected album and selected images are send here from the previous page. First we will find those information and then load image from Google. (LoadImage-method will be covered later in this article).

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

 // Find selected image index from parameters
 IDictionary<string, string> parameters = this.NavigationContext.QueryString;
 if (parameters.ContainsKey("SelectedIndex"))
 {
  app.selectedImageIndex = Int32.Parse(parameters["SelectedIndex"]);
 }
 else
 {
  app.selectedImageIndex = 0;
 }
 // Find selected album name
 if (parameters.ContainsKey("SelectedAlbum"))
 {
  selectedAlbumTitle = parameters["SelectedAlbum"];
 }
 else
 {
  selectedAlbumTitle = "No album name";
 }

 // Load image from Google
 LoadImage();
}
```

Silverlight Toolkit gives Gestures to Silverlight applications. Here we first initialize `GestureListener` in `ImagesPage()`-constructor and when drag is completed we will handle the event and display previous or next image in this album.

```
// Constructor
public ImagesPage()
{
 InitializeComponent();

 // Initialize GestureListener
 gestureListener = GestureService.GetGestureListener(ContentPanel);
 // Handle Dragging (to show next or previous image from Album)
 gestureListener.DragCompleted += new EventHandler<DragCompletedGestureEventArgs>(gestureListener_DragCompleted);
}
```

`gestureListener_DragCompleted()`-method will be called when drag is completed in screen. If horizontal change is more than zero, we have to show previous image - otherwise we load next image.

```
// Gesture - Drag is complete
void gestureListener_DragCompleted(object sender, DragCompletedGestureEventArgs e)
{
 // Left or Right
 if (e.HorizontalChange > 0)
 {
  // previous image (or last if first is shown)
  app.selectedImageIndex--;
  if (app.selectedImageIndex < 0) app.selectedImageIndex = app.albumImages.Count - 1;
 }
 else
 {
  // next image (or first if last is shown)
  app.selectedImageIndex++;
  if (app.selectedImageIndex > (app.albumImages.Count - 1)) app.selectedImageIndex = 0;
 }
 // Load image from Google
 LoadImage();
}
```

Above method will call `LoadImage()`-method to start loading image from Google Picasa service. All the image URL's are stored in `albumImages` collection in `App.xaml.cs` class. We are listening image loading and when image is loaded, we will remove Loading.... -text from the screen.

```
// Load Image from Google
private void LoadImage()
{
 // Load a new image
 bitmapImage = new BitmapImage(new Uri(app.albumImages[app.selectedImageIndex].content, UriKind.RelativeOrAbsolute));
 // Handle loading (hide Loading... animation)
 bitmapImage.DownloadProgress += new EventHandler<DownloadProgressEventArgs>(bitmapImage_DownloadProgress);
 // Loaded Image is image source in XAML
 image.Source = bitmapImage;
}
```

```
// Image is loaded from Google
void bitmapImage_DownloadProgress(object sender, DownloadProgressEventArgs e)
{
 // Hide loading... animation
 ShowProgress = false;
 // Disable LoadingListener for this image
 bitmapImage.DownloadProgress -= new EventHandler<DownloadProgressEventArgs>(bitmapImage_DownloadProgress);
 // Show image details in UI
 ImageInfoTextBlock.Text = String.Format("Album {0} : Image {1} of {2}.",
  selectedAlbumTitle,
  (app.selectedImageIndex+1),
  app.albumImages.Count);
}
```

Summary

(C) Copyright Nokia 2013. All rights reserved.

This code example shows how you can authenticate to Google services and how you can use different extensions in Windows Phone application. Hope you find this article useful and it helps you work with JSON in WP7.

You can download source codes here: File:PTMMyPicasa.zip

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.