Extending the Windows Phone Pictures Hub

This article explains how to extend the Windows Phone Pictures Hub with your application.

19 Aug 2012

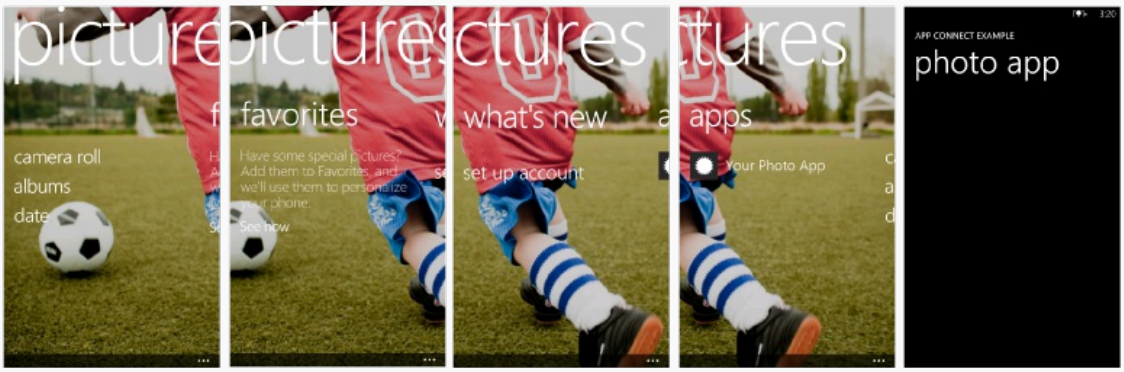
Introduction

The Pictures Hub is the centre of all still-image activity on Windows Phone. This article shows how an application can extend the Pictures Hub, adding itself as an "imaging" app that can be launched from the hub, picture viewer, or share picker option.

Note: This article extends the WP7 content part on MSDN (like Pictures Extensibility Overview for Windows Phone) with concrete examples and a downloadable example you can try out on your Windows Phone device.

Register your app as an "Imaging application"

The application can be registered in the manifest file as an *Imaging application*, it will then appear in the Pictures Hub "applications" section (as shown below in fourth-from-left image):



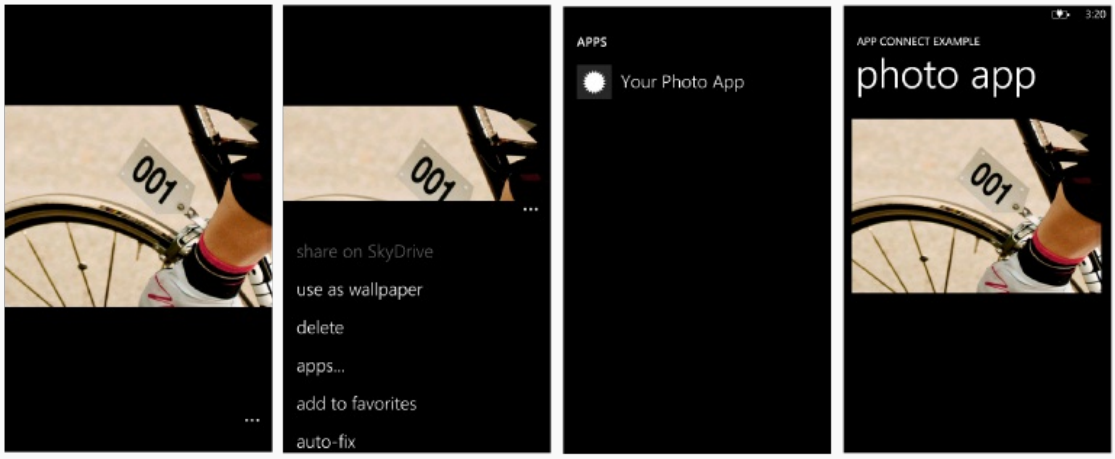
Pictures Hub with "Your Photo App" added. The right-most image shows the opened app

All that is required is to open the **WMAppManifest.xml** and add the following code fragment after the *Tokens* node:

```
<Extensions>
      <Extension ExtensionName="Photos_Extra_Hub"
                        ConsumerID="{5B04B775-356B-4AA0-AAF8-6491FFEA5632}"
                        TaskID="_default" />
</Extensions>
```

Register application as an image viewer

The application can be registered as an *Image viewer*. This allows the user to choose the app from the Pictures Hub's menu to open a selected image.



Opening an image using a registered image viewer

To register your app, first add the following text after the *Tokens node* in **WMAppManifest.xml**:

(C) Copyright Nokia 2013. All rights reserved.

```
<Extensions>
      <Extension ExtensionName="Photos_Extra_Viewer"
                        ConsumerID="{5B04B775-356B-4AA0-AAF8-6491FFEA5632}"
                        TaskID="_default" />
</Extensions>
```

In your code, override `OnNavigatedTo()` function. When your application is launched from Pictures hub, *token* is added in `NavigationContext.QueryString`. This token is used to get a `Picture` instance

```
using Microsoft.Xna.Framework.Media;
using System.Windows.Navigation;
using System.IO;

//...

protected override void OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e)
{
 IDictionary<string, string> queryStrings = this.NavigationContext.QueryString;
 //check if a token is added
 if (queryStrings.ContainsKey("token"))
 {

  MediaLibrary library = new MediaLibrary();
  //Acces to Picture instance from token.
  Picture picture = library.GetPictureFromToken(queryStrings["token"]);

  //Get Image Stream
  Stream imageStream = picture.GetImage()

  /*Todo  Code to open image and display it
   ...
  */
  queryStrings.Remove("token");

 }
 base.OnNavigatedTo(e);
}
```

Ensuring correct orientation of loaded image

Of course it's not so simple. When `BitmapImage` load image, it doesn't check image orientation. On Lumia 800 (maybe all WP7 devices) a portrait image will be loaded like a landscape Image. So you must check orientation manually. To do it you can use ExifLib library which can read exif image data and so **image orientation**.

Now you can easily perform correction image display.

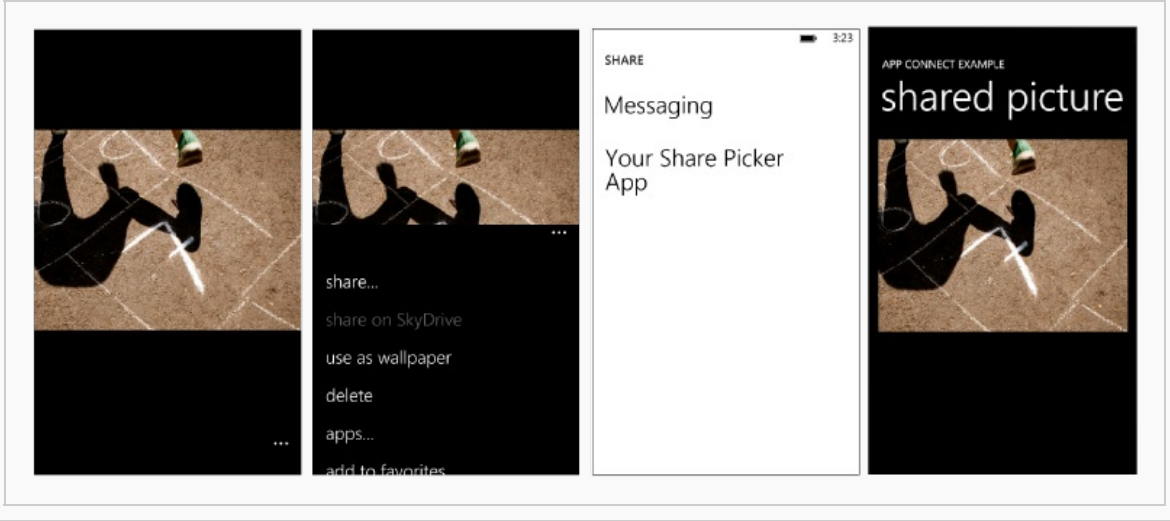
```
<Image x:Name="imageViewer" RenderTransformOrigin="0.5,0.5" HorizontalAlignment="Center" VerticalAlignment="Center" Margin="0,150,0,
    <Image.RenderTransform>
        <RotateTransform x:Name="imageOrientation" />
    </Image.RenderTransform>
</Image>
```

```
// Load Image to a BitmapImage
BitmapImage bitmap = new BitmapImage();
bitmap.CreateOptions = BitmapCreateOptions.None;
bitmap.SetSource(imageStream);

//display loaded image
imageViewer.Source = bitmap;
imageOrientation.Angle = 0;


//reposition stream to the beginning
imageStream.Seek(0, SeekOrigin.Begin);
//read exif data
JpegInfo info = ExifLib.ExifReader.ReadJpeg(imageStream, "");

if ( info.Orientation == ExifOrientation.TopRight)
{
 imageOrientation.Angle = 90;
}
if (info.Orientation == ExifOrientation.BottomRight)
{
 imageOrientation.Angle = 180;
}
if (info.Orientation == ExifOrientation.BottomLeft)
{
  imageOrientation.Angle = -90;
}
```

Register your app as an image sharer

The application can be registered as an **Image Sharer**, allowing it to be selected as an option for sharing an image (for example to a web service).

(C) Copyright Nokia 2013. All rights reserved.



Open **WMAppManifest.xml** and add the following text after the *Tokens* node.

```
<Extensions>
      <Extension ExtensionName="Photos_Extra_Share"
                        ConsumerID="{5B04B775-356B-4AA0-AAF8-6491FFEA5632}"
                        TaskID="_default" />
</Extensions>
```

In your code, override `OnNavigatedTo()` function. When your application is launched from Picture hub, `FileId` is added in `NavigationContext.QueryString`. This token is used to get a `Picture` instance

```
using Microsoft.Xna.Framework.Media;
using System.Windows.Navigation;
using System.IO;

//...

protected override void OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e)
{
 IDictionary<string, string> queryStrings = this.NavigationContext.QueryString;
 //check if a token is added
 if (queryStrings.ContainsKey("FileId"))
 {

  MediaLibrary library = new MediaLibrary();
  //Acces to Picture instance from token.
  Picture picture = library.GetPictureFromToken(queryStrings["FileId"]);

  //Get Image Stream
  Stream imageStream = picture.GetImage()

  /*Todo  your code
   ...
  */
  queryStrings.Remove("FileId");

 }
 base.OnNavigatedTo(e);
}
```

Reference links

ExifLib

- ExifLib
- Handling picture orientation in CameraCaptureTask in Windows Phone 7

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.