Accessing Instagram from your Windows Phone Application

This article explains how to consume the Instagram API from your Windows Phone Application

Introduction

Instagram is a popular online photo and video sharing social network. It allows its users to take pictures and videos, apply digital filters to them, and then share them with the community. At the moment there is no official Windows Phone support but there is an API that you can use to access the content that is shared by the users. *Note that the current API does not allow non-official clients to upload photos.*

This article will show you how to integrate Instagram in your application. For this example, we'll create a simple app to retrieve the list of your followers and the users you're following.

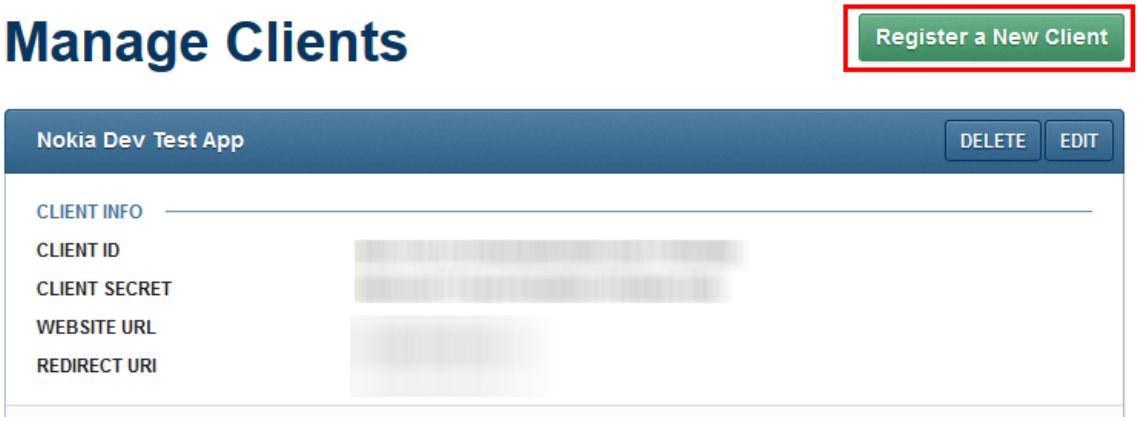
Prerequisites

To be able to access the Instagram API you must have an Instagram account. This account is free but can only be created using an official Instagram client, available for Android and iOS.

Getting your API key

First of all, you must be registered as a developer. You can register by filling the Developer Signup Form. Make sure you read their API Terms of Use!

In order to be able to use Instagram API, you must register your application to get an API Key. In this example we'll use http://instagram.com for the *Redirect URI*.



Take a note of your Client ID and Client Secret as we'll be needing them later.

Setting up the project

For this example we'll start with the Windows Phone Panorama App template and modify it to accustom our needs. Open a new instance of Visual Studio and create a **Windows Phone Panorama App** project.

Adding references

Next we'll add some NuGet packages that we will need:

- **Microsoft HTTP Client Libraries** (Microsoft.Net.Http) which contains the `HttpClient` class that will be used to make HTTP calls to the Instagram API
- **Json.NET** (Newtonsoft.Json) will help parse the JSON responses
- **Async for Windows Phone 7.5** (Microsoft.Bcl.Async) allows the use of the new async features (*required only for Windows Phone 7.5*)

To install NuGet packages you can go to the Solution Explorer, right click the project file and select **Manage NuGet Packages**. Search for the package name and click Install. You can also use the Package Manager Console to install these packages.

Modifying the Panorama template

Start by renaming the **ItemViewModel.cs** located under ViewModels to **UserViewModel.cs**. This class will hold the information for each Instagram user. In this article we'll just use Username, Full Name and Profile Picture.

(C) Copyright Nokia 2014. All rights reserved.

Open **UserViewModel.cs** and rename the `ItemViewModel` class to `UserViewModel`. Now change the properties of this class to `Username`, `FullName` and `ProfilePicUrl`. Your class should look similar to this:

```
public class UserViewModel : INotifyPropertyChanged
{
    private string _username;
    public string Username
    {
        get
        {
            return _username;
        }
        set
        {
            if (value != _username)
            {
                _username = value;
                NotifyPropertyChanged("Username");
            }
        }
    }

    private string _fullName;
    public string FullName
    {
        get
        {
            return _fullName;
        }
        set
        {
            if (value != _fullName)
            {
                _fullName = value;
                NotifyPropertyChanged("FullName");
            }
        }
    }

    private string _profilePicUrl;
    public string ProfilePicUrl
    {
        get
        {
            return _profilePicUrl;
        }
        set
        {
            if (value != _profilePicUrl)
            {
                _profilePicUrl = value;
                NotifyPropertyChanged("ProfilePicUrl");
            }
        }
    }
```


(C) Copyright Nokia 2014. All rights reserved.

```
    public event PropertyChangedEventHandler PropertyChanged;
    private void NotifyPropertyChanged(String propertyName)
    {
        PropertyChangedEventHandler handler = PropertyChanged;
        if (null != handler)
        {
            handler(this, new PropertyChangedEventArgs(propertyName));
        }
    }
}
```

Now open **MainViewModel.cs**. Change the `Items` property name to `Following`. This collection will hold all the users that you are currently following. Add another property named `Followers`, which will store all the users that are following you. You can also delete the `SampleProperty` and the sample items inside the `LoadData()` method. Your file should now look similar to this:

```
public class MainViewModel : INotifyPropertyChanged
{
    public MainViewModel()
    {

    }

    public ObservableCollection<UserViewModel> Following { get; private set; }
    public ObservableCollection<UserViewModel> Followers { get; private set; }


    public bool IsDataLoaded
    {
        get;
        private set;
    }

    public void LoadData()
    {
        // TODO

        this.IsDataLoaded = true;
    }

    public event PropertyChangedEventHandler PropertyChanged;
    private void NotifyPropertyChanged(String propertyName)
    {
        PropertyChangedEventHandler handler = PropertyChanged;
        if (null != handler)
        {
            handler(this, new PropertyChangedEventArgs(propertyName));
        }
    }
}
```

Now we need to change the UI to reflect this changes. Open the **MainPage.xaml** and change your Panorama control to the following:

```
<!--Panorama control-->
```


(C) Copyright Nokia 2014. All rights reserved.

```
<controls:Panorama Title="instagram sample">
    <controls:Panorama.Background>
        <ImageBrush ImageSource="PanoramaBackground.png"/>
    </controls:Panorama.Background>

    <!--Panorama item one-->
    <controls:PanoramaItem Header="following">
        <!--Double line list with text wrapping-->
        <ListBox Margin="0,0,-12,0" ItemsSource="{Binding Following}">
            <ListBox.ItemTemplate>
                <DataTemplate>
                    <StackPanel Orientation="Horizontal" Margin="0,0,0,17">
                        <Image Height="100" Width="100" Margin="12,0,9,0"
Source="{Binding ProfilePicUrl}" />
                        <StackPanel Width="311">
                            <TextBlock Text="{Binding Username}" TextWrapping="Wrap"
Style="{StaticResource PhoneTextExtraLargeStyle}"/>
                            <TextBlock Text="{Binding FullName}" TextWrapping="Wrap"
Margin="12,-6,12,0" Style="{StaticResource PhoneTextSubtleStyle}"/>
                        </StackPanel>
                    </StackPanel>
                </DataTemplate>
            </ListBox.ItemTemplate>
        </ListBox>
    </controls:PanoramaItem>

    <!--Panorama item two-->
    <!--Use 'Orientation="Horizontal"' to enable a panel that lays out horizontally-->
    <controls:PanoramaItem Header="followers">
        <!--Double line list with image placeholder and text wrapping-->
        <ListBox Margin="0,0,-12,0" ItemsSource="{Binding Followers}">
            <ListBox.ItemTemplate>
                <DataTemplate>
                    <StackPanel Orientation="Horizontal" Margin="0,0,0,17">
                        <Image Height="100" Width="100" Margin="12,0,9,0"
Source="{Binding ProfilePicUrl}" />
                        <StackPanel Width="311">
                            <TextBlock Text="{Binding Username}" TextWrapping="Wrap"
Style="{StaticResource PhoneTextExtraLargeStyle}"/>
                            <TextBlock Text="{Binding FullName}" TextWrapping="Wrap"
Margin="12,-6,12,0" Style="{StaticResource PhoneTextSubtleStyle}"/>
                        </StackPanel>
                    </StackPanel>
                </DataTemplate>
            </ListBox.ItemTemplate>
        </ListBox>
    </controls:PanoramaItem>
</controls:Panorama>
```

Finally open **MainViewModelSampleData.xaml** under the **SampleData** folder. This file holds the sample data that populates the UI during design time. Since we changed our view models we must also modify the sample data to reflect our changes.

We'll be using the default Application Icon as a Profile Picture for your users. Replace the existing data with the following:

```
    <local:MainViewModel.Following>
        <local:UserViewModel Username="user 1" FullName="User 1 Full Name"
```


(C) Copyright Nokia 2014. All rights reserved.

```
ProfilePicUrl="/ApplicationIcon.png"/>
        <local:UserViewModel Username="user 2" FullName="User 2 Full Name"
ProfilePicUrl="/ApplicationIcon.png"/>
        <local:UserViewModel Username="user 3" FullName="User 3 Full Name"
ProfilePicUrl="/ApplicationIcon.png"/>
        <local:UserViewModel Username="user 4" FullName="User 4 Full Name"
ProfilePicUrl="/ApplicationIcon.png"/>
        <local:UserViewModel Username="user 5" FullName="User 5 Full Name"
ProfilePicUrl="/ApplicationIcon.png"/>
        <local:UserViewModel Username="user 6" FullName="User 6 Full Name"
ProfilePicUrl="/ApplicationIcon.png"/>
    </local:MainViewModel.Following>

    <local:MainViewModel.Followers>
        <local:UserViewModel Username="user A" FullName="User A Full Name"
ProfilePicUrl="/ApplicationIcon.png"/>
        <local:UserViewModel Username="user B" FullName="User B Full Name"
ProfilePicUrl="/ApplicationIcon.png"/>
        <local:UserViewModel Username="user C" FullName="User C Full Name"
ProfilePicUrl="/ApplicationIcon.png"/>
        <local:UserViewModel Username="user D" FullName="User D Full Name"
ProfilePicUrl="/ApplicationIcon.png"/>
        <local:UserViewModel Username="user E" FullName="User E Full Name"
ProfilePicUrl="/ApplicationIcon.png"/>
        <local:UserViewModel Username="user F" FullName="User F Full Name"
ProfilePicUrl="/ApplicationIcon.png"/>
    </local:MainViewModel.Followers>
```

Also, be sure to remove the line:

```
SampleProperty="Sample Text Property Value"
```

Thanks to this sample data, you can see how your application will look like when populated with data:

(C) Copyright Nokia 2014. All rights reserved.

Using Instagram API

All the API endpoints are located at **api.instagram.com** and are only accessible via http**s**.

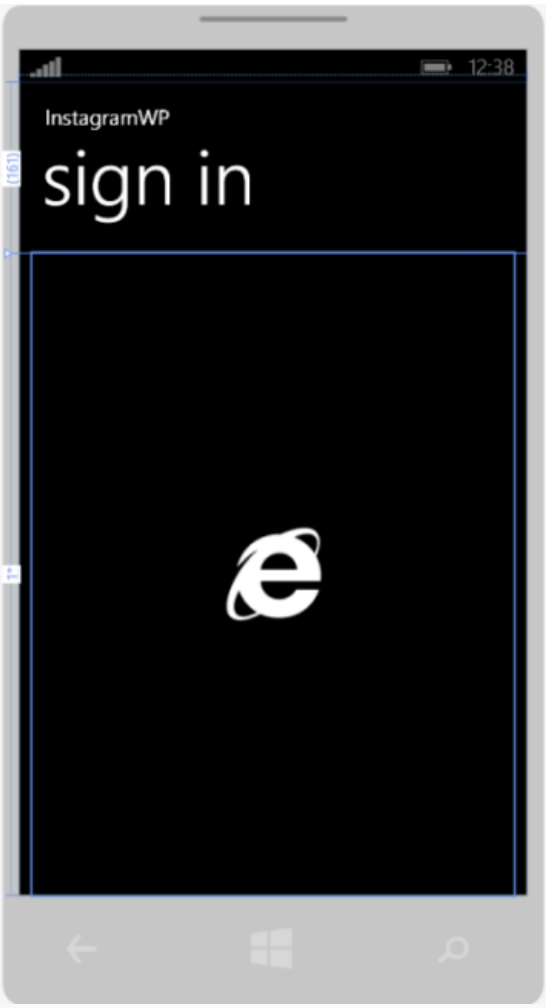
Be sure to check Instagram Developer Documentation.

OAuth Authentication

In order to be able to access the Instagram API you must first authenticate using the OAuth 2.0 protocol. For more information about this protocol see the current draft. You can also check this article: OAuth on Windows Phone.

Add a new Blank Page and name it Authentication.xaml. Add a `WebBrowser` control and name it `AuthBrowser`. Add the `Navigated()` event handler to the `AuthBrowser`. We will use this `WebBrowser` control to obtain an access token that we can use to access the API.

(C) Copyright Nokia 2014. All rights reserved.




```
<phone:WebBrowser x:Name="AuthBrowser"
Navigated="AuthBrowser_Navigated" />
```

```
protected override void OnNavigatedTo(NavigationEventArgs e)
{
 base.OnNavigatedTo(e);
 AuthBrowser.Navigate(new Uri("https://instagram.com/oauth/authorize/?
client_id=<YOUR_CLIENT_ID>&redirect_uri=http://instagram.com&response_type=token"));
}

void AuthBrowser_Navigated(object sender, NavigationEventArgs e)
{
 //access token is a Url fragment and these fragments start with '#'
 if (e.Uri.AbsoluteUri.Contains('#'))
 {
  //parse our access token
  if (e.Uri.Fragment.StartsWith("#access_token="))
  {
   string token = e.Uri.Fragment.Replace("#access_token=", string.Empty);

   //save our token
   IsolatedStorageSettings.ApplicationSettings["access_token"] = token;
   IsolatedStorageSettings.ApplicationSettings.Save();

   //now that we have our token, let's go back to the MainPage
   NavigationService.GoBack();
  }
 }
}
```


(C) Copyright Nokia 2014. All rights reserved.

Make sure you replace **YOUR_CLIENT_ID** with your own Client ID.

By default all apps have read access. If you wish to use extended access, such as liking, commenting, or managing friendships you'll need to add the **scope** parameter to your request url. According to the Instagram API documentation, these are the currently supported scopes:

- **basic** - to read any and all data related to a user (e.g. following/followed-by lists, photos, etc.) (granted by default)
- **comments** - to create or delete comments on a user's behalf
- **relationships** - to follow and unfollow users on a user's behalf
- **likes** - to like and unlike items on a user's behalf

You should only request the scope you need at the time of authorization. If in the future you require additional scope, you may forward the user to the authorization URL with that additional scope to be granted. If you attempt to perform a request with an access token that isn't authorized for that scope, you will receive an `OAuthPermissionsException` error return.

If you'd like to request multiple scopes at once, simply separate the scopes by a space. In the url, this equates to an escaped space ("+"). So if you're requesting the likes and comments permission, the parameter will look like this: *scope=likes+comments*

Note that an empty scope parameter (scope=) is invalid; you must either omit the scope, or specify a non-empty scope list.

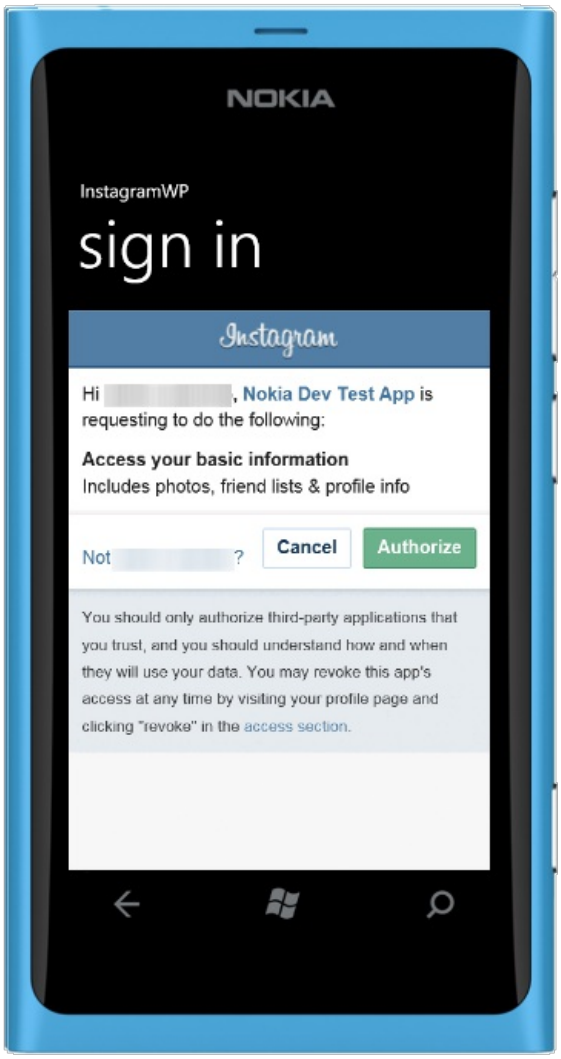
Finally go to **MainPage.xaml.cs**. Change the `MainPage_Loaded` event handler so that you can redirect the user to the `Authentication` view you created.

```
private void MainPage_Loaded(object sender, RoutedEventArgs e)
{

 if (IsolatedStorageSettings.ApplicationSettings.Contains("access_token"))
 {
  if (!App.ViewModel.IsDataLoaded)
  {
   App.ViewModel.LoadData();
  }
 }
 else
  NavigationService.Navigate(new Uri("/Authentication.xaml", UriKind.Relative));
        }
```

Try and run your app. The authentication process will look like this:

(C) Copyright Nokia 2014. All rights reserved.



Accessing content

Now that we have an access token, we can start making API calls. As mentioned earlier, for this example, we'll get the list of followers and the list of people that we're following.

Create a class named **InstagramAPI.cs**. In this file we'll put all the code required to use the API.

```
using Newtonsoft.Json;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Net;
using System.Net.Http;
using System.Threading.Tasks;

public class InstagramAPI
{
 private readonly string apiBaseUrl = "https://api.instagram.com/v1";
 private readonly string apiAccessToken;

 private HttpClient _httpClient;

 public InstagramAPI(string accessToken)
 {
  this.apiAccessToken = accessToken;
  _httpClient = new HttpClient();
 }
```


(C) Copyright Nokia 2014. All rights reserved.

```
 private async Task<T> SendAsync<T>(HttpRequestMessage request) where T : class
 {

  HttpResponseMessage response;
  response = await _httpClient.SendAsync(request);

  //TODO: Error Handling
  if (response.StatusCode == HttpStatusCode.OK)
  {
   string responseBody = await response.Content.ReadAsStringAsync();

   return JsonConvert.DeserializeObject<T>(responseBody);
  }

  return null;
 }

 public async Task<ObservableCollection<UserViewModel>> GetFollowers()
 {
  var request = new HttpRequestMessage(HttpMethod.Get, apiBaseUrl +
"/users/self/followed-by?access_token=" + apiAccessToken);
  var followersResponse =  await SendAsync<InstagramUserResponse>(request);
  var result = new ObservableCollection<UserViewModel>();
  foreach (var u in followersResponse.data)
  {
   result.Add(new UserViewModel() { FullName = u.full_name, Username = u.username,
ProfilePicUrl = u.profile_picture });
  }
  return result;
 }

 public async Task<ObservableCollection<UserViewModel>> GetFollowing()
 {
  var request = new HttpRequestMessage(HttpMethod.Get, apiBaseUrl +
"/users/self/follows?access_token=" + apiAccessToken);
  var followingResponse = await SendAsync<InstagramUserResponse>(request);
  var result = new ObservableCollection<UserViewModel>();
  foreach (var u in followingResponse.data)
  {
   result.Add(new UserViewModel() { FullName = u.full_name, Username = u.username,
ProfilePicUrl = u.profile_picture });
  }
  return result;
 }
}

public class InstagramUser
{
 public string username { get; set; }
 public string bio { get; set; }
 public string website { get; set; }
 public string profile_picture { get; set; }
 public string full_name { get; set; }
 public string id { get; set; }
}

public class InstagramResponse
```


(C) Copyright Nokia 2014. All rights reserved.

```
{
 public List<InstagramUser> data { get; set; }
}
```

The `SendAsync` method is a generic method that can be used to send a request and return a specific type of object. It uses the **Json.Net** package to parse the JSON response and map it to a .Net object, `InstagramUser` and `InstagramResponse` in this example.

Displaying the content

Finally we need to display the data in the UI. Go to the **MainViewModel.cs** file and locate the `LoadData()` method. Add the following code:

```
public async void LoadData()
{
 string accessToken =
(string)IsolatedStorageSettings.ApplicationSettings["access_token"];

 InstagramAPI apiClient = new InstagramAPI(accessToken);
 Following = await apiClient.GetFollowing();
 NotifyPropertyChanged("Following");
 Followers =  await apiClient.GetFollowers();
 NotifyPropertyChanged("Followers");
 this.IsDataLoaded = true;
}
```

That's it, you should now be able to see the users you are following and a list of people that follow you!

Summary

This should give you an idea on how easy is to access Instagram from your Windows Phone application. Be sure to check the Instagram API documentation to understand the various features that you can integrate into your application.

You can download the code for this article here: Media:InstagramWP7.zip. It is a Windows Phone 7.5 project that will run on any Windows Phone 7 or 8 device. Happy coding!

(C) Copyright Nokia 2014. All rights reserved.