Como utilizar a API do Instagram em uma aplicação Windows Phone

Introdução

O Instagram é uma popular rede social de partilha de imagens e curtos vídeos. Permite aos utilizadores capturar fotos ou vídeos, aplicar filtros digitais e partilhar com a comunidade. De momento não existe um cliente oficial para Windows Phone. No entanto, existe uma API que pode ser utilizada para ter acesso ao conteúdo compartilhado pelos usuários desse serviço. Este artigo explica como utilizar a API do Instagram a partir da tua Aplicação Windows Phone.

Este artigo demonstra como integrar conteúdos do Instagram na tua aplicação. Para este exemplo iremos criar uma simples app que obtém a lista de seguidores e os utilizadores que estás a seguir no Instagram.

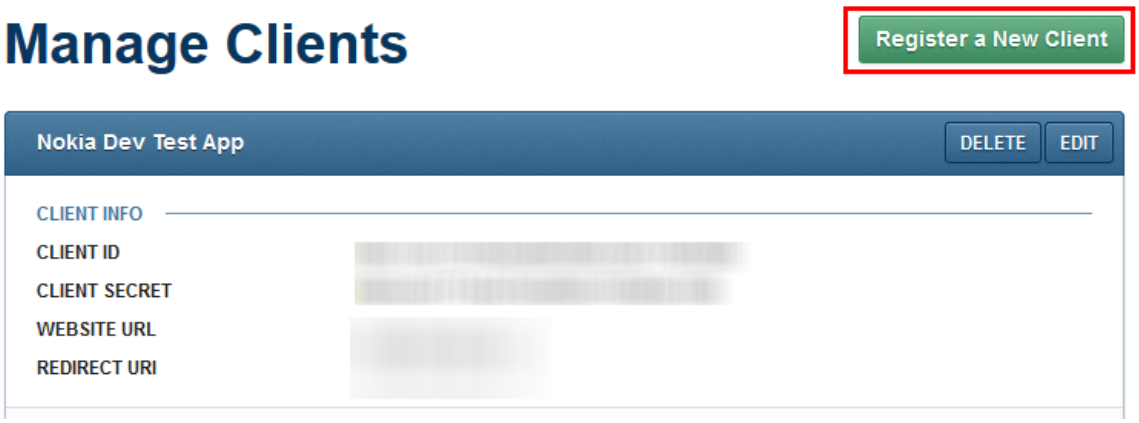
Pré-requisitos

Para utilizar a API é necessário uma conta Instagram. Esta conta é gratuita mas pode apenas ser criada recorrendo a uma aplicação oficial do Instagram, disponível apenas para Android e iOS.

Obtendo acesso à API

Em primeiro lugar deves estar registado como developer. Este registo pode ser feito preenchendo o Formulário de Registo. Certifica-te que lês os Termos de Uso da API!

Para utilizar a API é necessário registar a aplicação que estás a desenvolver de forma a obter uma chave de acesso à API. De forma a facilitar com que consigas seguir este exemplo, certifica-te que colocas **http://instagram.com** no campo **Redirect URI** ao preencher o formulário.



Convém tomar nota do Client ID, será necessário mais tarde.

Criação do projeto

Para este exemplo será usado o template Windows Phone Panorama App, que será modificado de acordo com as necessidades.

Adicionando referências

De seguida vamos adicionar alguns pacotes recorrendo ao gestor NuGet. Os pacotes são os seguintes:

- **Microsoft HTTP Client Libraries** (Microsoft.Net.Http) contém a classe HttpClient que será usada para fazer os pedidos http à API do Instagram;
- **Json.NET** (Newtonsoft.Json) irá ajudar a traduzir as respostas JSON em objectos .Net
- **Async for Windows Phone 7.5** (Microsoft.Bcl.Async) permite a utilização das novas funcionalidades *async* (*apenas necessário para Windows Phone 7.5*)

Para instalar estes pacotes basta, no Solution Explorer, clicar com o botão direito no Projecto e seleccionar **Manage NuGet Packages**. Uma simples pesquisa deve apresentar o pacote pretendido, bastando depois carregar em Install. Também é possível instalar estes pacotes recorrendo à Package Manager Console.

Modificando o template

(R) Copyright Nokia 2014. All rights reserved.

Começamos por mudar o nome do ficheiro **ItemViewModel.cs** na pasta ViewModels para **UserViewModel.cs**. Esta classe serve para armazenar a informação relativa a um utilizador Instagram. Neste artigo usaremos como exemplo 3 campos: Username, Nome e Imagem de Perfil. Depois, no ficheiro **UserViewModel.cs** muda-se o nome da classe *ItemViewModel* para *UserViewModel*. Alteramos também as propriedades para Username, FullName e ProfilePicUrl. Após as alterações a classe deve ter este aspeto:

```
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");
            }
```


(R) 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));
        }
    }
}
```

Vamos agora ao ficheiro **MainViewModel.cs**. Renomeamos a propriedade 'Items *para* Following*. Esta coleção será usada para armazenar os utilizadores que actualmente seguimos. Adicionamos outra propriedade chamada* Followers*, usada para armazenar a coleção de utilizadores que nos estão a seguir. A propriedade* SampleProperty *pode ser apagada e o corpo do método* LoadData() *pode ser apagado.*

```
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));
        }
    }
}
```

Agora é necessário alterar o UI de modo a reflectir estas alterações. No ficheiro **MainPage.xaml** vamos aplicar as seguintes

(R) Copyright Nokia 2014. All rights reserved.

alterações ao Panorama:

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

Por fim vamos modificar o ficheiro **MainViewModelSampleData.xaml** na pasta SampleData folder. Este ficheiro contém dados exemplo que preenchem o UI em design time. Uma vez que alteramos os viewmodels, é também necessário modificar os dados exemplo para reflectir estas alterações. A título de exemplo vamos utilizar o Application Icon padrão como Imagem de Perfil.

(R) Copyright Nokia 2014. All rights reserved.

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

A linha *SampleProperty="Sample Text Property Value"* deve também ser removida, uma vez que esta propriedade foi eliminada do nosso modelo.

Graças a estes dados exemplo, é possível visualizar o aspecto da aplicação quando estiver preenchida com informação.

(c) Copyright Nokia 2014. All rights reserved.

Utilizando a API Instagram

Todos os endpoints da API encontram-se em **api.instagram.com** e devem apenas ser acedidos usando o protocolo http**s**.

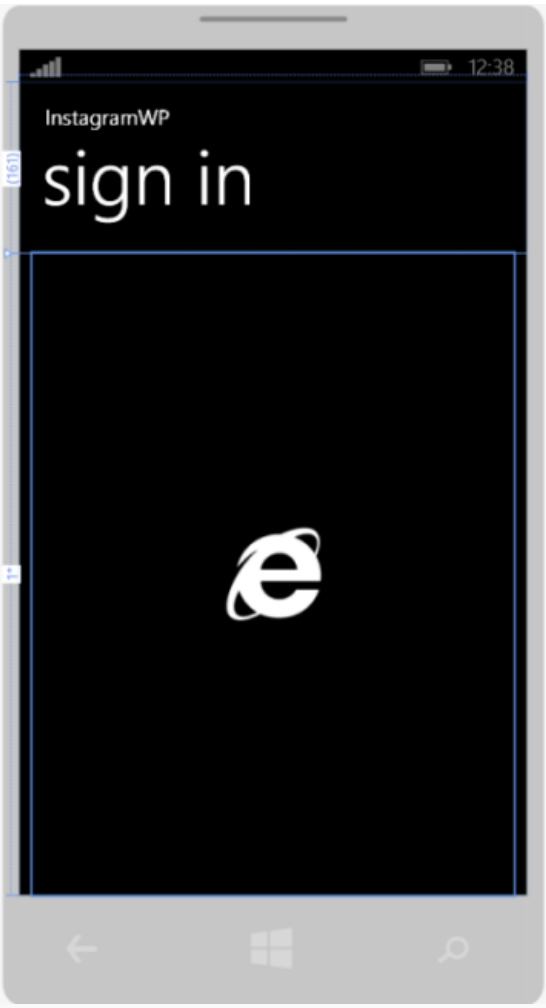
Recomenda-se a leitura da Instagram Developer Documentation.

Autenticação OAuth

Cada pedido à API deve ser autenticado com um token de acesso. Esta autenticação é feita recorrendo ao protocolo OAuth 2.0. Para mais informações sobre este protocolo, consultar a especificação. Recomenda-se também a leitura do seguinte artigo: OAuth on Windows Phone (em inglês).

Adicionamos uma nova Blank Page com o nome Authentication.xaml. A esta página adicionamos um controlo **WebBrowser** com o nome AuthBrowser. Adicionamos também o event handler para o evento **Navigated**. Este WebBrowser será usado para obter um token de acesso à 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)
{
 //o token de acesso é devolvido como um fragmento no url, que começa por '#'
 if (e.Uri.AbsoluteUri.Contains('#'))
 {
  //obter o token do fragmento
  if (e.Uri.Fragment.StartsWith("#access_token="))
  {
   string token = e.Uri.Fragment.Replace("#access_token=", string.Empty);

   //guardar o token
   IsolatedStorageSettings.ApplicationSettings["access_token"] = token;
   IsolatedStorageSettings.ApplicationSettings.Save();

   //agora que temos o token, voltar para a página principal
   NavigationService.GoBack();
  }
 }
}
```


(R) Copyright Nokia 2014. All rights reserved.

Não esquecer de substutuir **<YOUR_CLIENT_ID>** pelo Client ID que foi devolvido quando foi registada a aplicação no site do Instagram.

Por defeito todas as aplicações que acedem à API têm apenas permissões de leitura. Caso seja necessário um acesso mais amplo, como comentar ou gostar de fotos ou gerir amizades, é necessário adicionar o parâmetro **scope** ao url do pedido. De acordo com a Instagram API documentation, estas são as permissões extra suportadas:

- **basic** - para ler qualquer tipo de informação relacionada com o utilizador (e.g. listas de seguidores/utilizadores seguidos, fotos, etc.) (garantida por defeito)
- **comments** - para criar ou remover comentários em nome do utilizador
- **relationships** - para seguir ou deixar de seguir utilizadores
- **likes** - para gostar ou desgostar de items

Devem apenas ser requeridas estar permissões aquando da autorização. No futuro, caso sejam necessárias permissões adicionais, o utilizador deve ser encaminhado para o url de autorização com as permissões adicionais a serem garantidas. Caso seja feito um pedido com um token de acesso com permissões fora desse âmbito é devolvido um erro OAuthPermissionsException.

Caso a intenção seja pedir acesso a vários scopes em simultâneo, estes devem ser separados por um espaço. No url isto equivale a um '+'. Portanto, para pedir permissões para gostos e comentários, por exemplo, o parâmetro terá o seguinte aspecto: *scope=likes+comments*

De salientar que o parametro scope vazio (scope=) é inválido; deve ser omitido ou deve ser especificada uma lista não vazia.

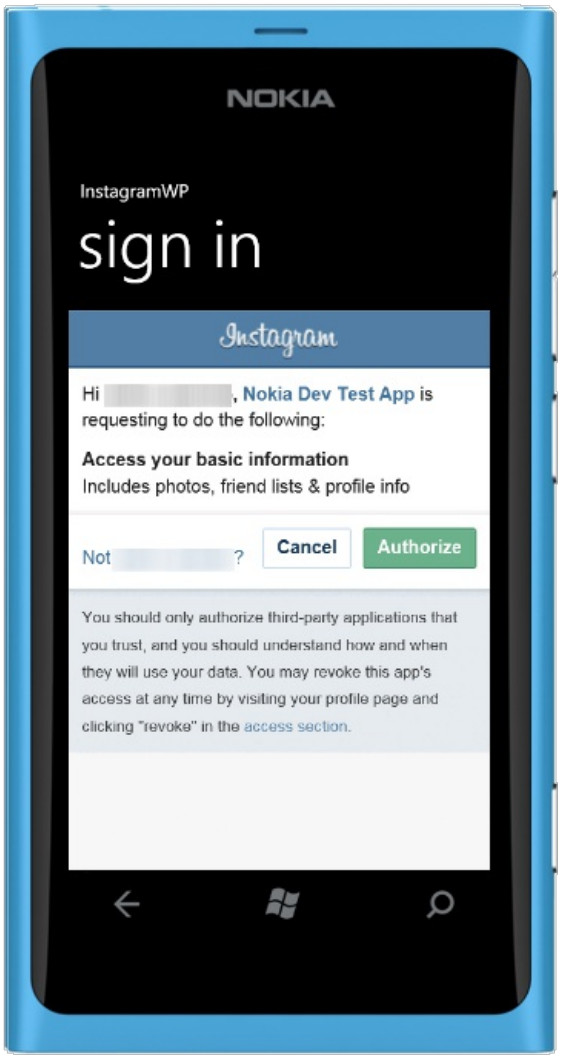
Por fim abrimos o ficheiro **MainPage.xaml.cs** e alteramos o event handler **MainPage_Loaded** de forma a redireccionar o utilizador para a página de Autenticação caso não exista um token de acesso à API.

```
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));
        }
```

Caso tentemos correr a aplicação neste momento, este será o aspecto do processo de autenticação:

(R) Copyright Nokia 2014. All rights reserved.



Acessando conteúdos

Agora que temos um token de acesso podemos começar a fazer pedidos à API. Como mencionado na introdução, para este exemplo, vamos obter a lista de seguidores e de pessoas que estamos a seguir.

Vamos criar uma classe chamada **InstagramAPI.cs**. Será neste ficheiro que colocaremos o código necessário para utilizar a 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; }
}
```


(R) Copyright Nokia 2014. All rights reserved.

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

O método *SendAsync* é um método genérico que pode ser usado para enviar um pedido e devolver um objecto com um tipo específico. Recorre ao pacote Json.Net para fazer a tradução da resposta em JSON e mapear esta resposta para um objecto .Net, neste caso InstagramUser e InstagramResponse.

Apresentando o conteúdo

Por fim é necessário apresentaro conteúdo que obtemos da API no nosso User Interface. Para isto, vamos ao ficheiro **MainViewModel.cs**, mais especificamente ao método **LoadData()**. Aqui adicionamos o seguinte código:

```
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;
}
```

E pronto! É agora possível visualizar os utilizadores que seguimos e os utilizadores que nos estão a seguir.

Sumário

Este artigo deve dar uma ideia de quão fácil é acessar conteúdo da rede Instagram a partir de uma aplicação Windows Phone. A documentação da API do Instagram deve ser consultada para se entender as várias características que podem ser utilizadas na tua aplicação. O código deste artigo pode ser obtido aqui: Media:InstagramWP7.zip. Consiste num projeto Windows Phone 7.5 que pode ser executado em qualquer dispositivo Windows Phone 7 ou 8. Bom trabalho!

(c) Copyright Nokia 2014. All rights reserved.