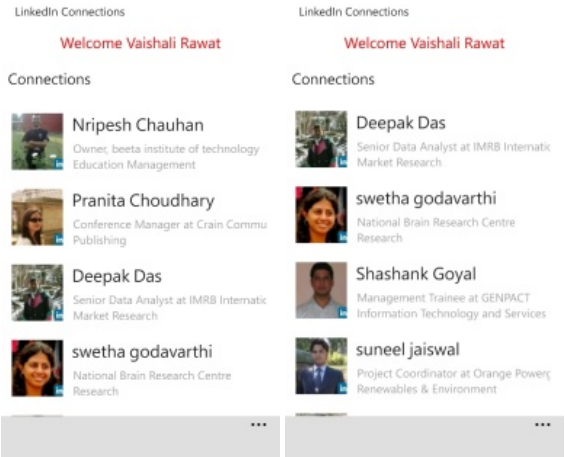
# LinkedIn: Viewing Connections using Hammock

This article explains how to fetch the logged in user's 1st degree LinkedIn's connections list using the Hammock library.

LinkedIn uses OAuth authentication process, implementing which is a trivial task as it requires a lot of stuff generation like signature key, signature method key, nonce key, timestamp key etc. To getting rid of all this work, we have a library named Hammock which is a .Net library that simplifies consuming and wrapping RESTful services. So, in this article we will use Hammock library to make all the REST server hits.

## Introduction

In this article, we will fetch the current user's 1st degree connections list and will display them in a list box.




**Note:** This article also showcases how to parse both JSON and XML data.

## Prerequisites

User should be already logged in to LinkedIn to use the code shown in this article. You may refer LinkedIn: Authentication and viewing own basic profile article to know about how to login to LinkedIn site using Hammock.

**Note:** In our previous LinkedIn: Authentication and viewing own basic profile article, we didn't ask for any special permission from the user. However, in this article, we ask for the permission from the user to fetch the connections list. So, here we will ask for **r_network** permission.

## References Required

Following references to the DLLs are required in order to implement JSON parsing.

- System.RunTime.Serialization
- System.Servicemodel
- System.Servicemodel.Web

These can be added by Project|| References || Add Reference..

## Creating UI

Our UI consists of a single screen in which controls would be made visible/invisible depending on requirement. Initially we will display a web browser control which would be mapped to LinkedIn login page. After the login process, we will hide the web browser control, and display a text box with the logged in user's name, another text box for writing "Connections" heading and finally a list box. Each of the list item will showcase user's profile pic, his/her name, profile headline and the Industry as per mentioned in his/her LinkedIn profile.

The code in .xaml file is as shown below.

```
        <!--TitlePanel contains the name of the application and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle" Text="LinkedIn Connections" Style="{StaticResource PhoneTextNormalStyle}"/>
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <Grid.RowDefinitions>
                <RowDefinition Height=".05*"/>
                <RowDefinition Height="*"/>
            </Grid.RowDefinitions>
```


(C) Copyright Nokia 2013. All rights reserved.

```
            <phone:WebBrowser Grid.Row="0" Grid.RowSpan="2" Margin="-6,3,0,1" Name="loginBrowserControl" Visibility="Collapsed"
                              Navigated="loginBrowserControl_Navigated" Navigating="loginBrowserControl_Navigating"
                              IsScriptEnabled="True"/>

            <TextBlock Grid.Row="0" Visibility="Collapsed" x:Name="txtWelcomeText" VerticalAlignment="Top" HorizontalAlignment="Cent

            <Grid x:Name="UserPanel" Grid.Row="2" Visibility="Collapsed">
                <Grid.RowDefinitions>
                    <RowDefinition Height=".16*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>

                <TextBlock Grid.Row="0" Text="Connections" VerticalAlignment="Center" HorizontalAlignment="Left" FontSize="26"/>
                <ListBox x:Name="connectionsList" Grid.Row="1">
                    <ListBox.ItemTemplate>
                        <DataTemplate>
                            <Grid Height="130">
                                <Grid.ColumnDefinitions>
                                    <ColumnDefinition Width="100"/>
                                    <ColumnDefinition Width="*"/>
                                </Grid.ColumnDefinitions>
                                <!--<Image Source="{Binding PictureUrl}"-->
                                <Image delay:LowProfileImageLoader.UriSource="{Binding PictureUrl}"
                                Grid.Column="0"
                                   Width="97"
                                   Height="125"
                                   VerticalAlignment="Center"
                                   HorizontalAlignment="Center"
                                   Margin="6"/>
                                <StackPanel Margin="10,15,0,0"
                                        Grid.Column="1"
                                        Height="60"
                                        Orientation="Horizontal"
                                        VerticalAlignment="Top">
                                    <TextBlock Text="{Binding FirstName}"
                                           FontSize="30" />
                                    <TextBlock Text=" " />
                                    <TextBlock Text="{Binding LastName}"
                                           FontSize="30"/>
                                </StackPanel>
                                <StackPanel Margin="0,50,0,0"
                                        Grid.Column="1"
                                        VerticalAlignment="Center">
                                    <TextBlock Grid.Column="1"
                                       Text="{Binding Headline}"
                                       Style='{StaticResource PhoneTextSubtleStyle}'
                                       />
                                    <TextBlock Grid.Column="1"
                                       Text="{Binding Industry}"
                                       Style='{StaticResource PhoneTextSubtleStyle}'
                                       />
                                </StackPanel>
                            </Grid>
                        </DataTemplate>
                    </ListBox.ItemTemplate>
                </ListBox>
            </Grid>
        </Grid>
    </Grid>
    <!--Sample code showing usage of ApplicationBar-->
    <phone:PhoneApplicationPage.ApplicationBar>
        <shell:ApplicationBar IsVisible="True" IsMenuEnabled="True">
            <shell:ApplicationBar.MenuItems>
                <shell:ApplicationBarMenuItem Text="Sign In"  Click="MenuItemSignIn_Click"/>
                <shell:ApplicationBarMenuItem Text="Sign Out" Click="MenuItemSignOut_Click" IsEnabled="False"/>
            </shell:ApplicationBar.MenuItems>
        </shell:ApplicationBar>
    </phone:PhoneApplicationPage.ApplicationBar>
```

- We have bound the values of text boxes and image with their corresponding values. These values are declared in one of our used file as explained in the later part of the article.

## Files Used

In this article, we've used a few more files as compared to the previous article. The files used in this article are:

- AppSettings.cs - Same as described in previous article
- MainUtil.cs - Same as described in previous article
- SelfPerson.cs - This file will be used to parse the response received when we will try to access logged in user's profile.
- person.cs - This file will be used to parse the response received when we try to fetch connections list.
- Connections.cs - It will be used in JSON parsing.

### SelfPerson.cs File's contents

This file has following declaration.

```
    [XmlRoot("person")]
     public class SelfPerson
     {
         [XmlElement("first-name")]
         public string SelfFirstName { get; set; }

         [XmlElement("last-name")]
         public string SelfLastName { get; set; }
     }
```

### person.cs File's contents

This file has the following declaration.


(C) Copyright Nokia 2013. All rights reserved.

```
    [DataContract]
    public class person
    {
        [DataMember(Name = "firstName")]
        public string FirstName { get; set; }

        [DataMember(Name = "headline")]
        public string Headline { get; set; }

        [DataMember(Name = "id")]
        public string ID { get; set; }

        [DataMember(Name = "industry")]
        public string Industry { get; set; }

        [DataMember(Name = "lastName")]
        public string LastName { get; set; }

        [DataMember(Name = "pictureUrl")]
        public string PictureUrl { get; set; }
    }
```

## Connections.cs File's contents

This file has the following declaration.

```
  [DataContract]
    public class Connections
    {
        [DataMember(Name = "_total")]
        public string TotalConnections { get; set; }

        [DataMember(Name = "values")]
        public person[] ConnectionsList { get; set; }
    }
```

## MainPage.cs File's contents

In addition to the code explained in the previous article, this file has the following functions.

```
  private void fetchConnections()
        {
            var credentials = new OAuthCredentials
            {
                Type = OAuthType.ProtectedResource,
                SignatureMethod = OAuthSignatureMethod.HmacSha1,
                ParameterHandling = OAuthParameterHandling.UrlOrPostParameters,
                ConsumerKey = AppSettings.consumerKey,
                ConsumerSecret = AppSettings.consumerKeySecret,
                Token = this.accessToken,
                TokenSecret = this.accessTokenSecret,
                Version = "1.0"
            };

            var restClient = new RestClient
            {
                Authority = "http://api.linkedin.com",
                HasElevatedPermissions = true
            };

            var restRequest = new RestRequest
            {
                Credentials = credentials,
                Path = "/v1/people/~/connections:(first-name,headline,id,industry,last-name,pictureUrl)?format=json",
                Method = WebMethod.Get,
                Encoding = Encoding.UTF8
            };

            restClient.BeginRequest(restRequest, new RestCallback(ConnectionsRequestCallback));
        }
```

As shown above, we are setting the *path* value in the Hammock's `RestRequest()` as a combination of the actual required URL; the parameters whose values we want and the required response format type. If we don't mention the required params here then by default we will get all the attributes in our response. By providing *json* in the *format*, we are requesting that we want our response in JSON format. By default it returns the result in XML format.

On receiving the response successfully, we will parse it as shown below:

```
 private void ConnectionsRequestCallback(RestRequest request, RestResponse response, object obj)
        {
            Deployment.Current.Dispatcher.BeginInvoke(() =>
            {
                if (response.StatusCode == HttpStatusCode.OK)
                {
                    string result = response.Content.ToString();
                    parseConnectionsData(result);
                }
                else
                {
                    MessageBox.Show("failed");
                }
            });
        }

        private void parseConnectionsData(string aRespString)
        {
            connectionsObj = new Connections();
            MemoryStream ms = new MemoryStream(Encoding.UTF8.GetBytes(aRespString));
            DataContractJsonSerializer ser = new DataContractJsonSerializer(connectionsObj.GetType());
            connectionsObj = ser.ReadObject(ms) as Connections;
            ms.Close();

            // updating UI
            if (connectionsObj != null)
```


(C) Copyright Nokia 2013. All rights reserved.

```
            {
                updateList();
            }
        }
```

On successfully parsing the data, we need to update our UI. For updating the UI, we could have simply set the `ListBox`'s source as our **Connections** object. But this can be a bit problematic as our response doesn't always have all the keys and hence their values. Due to this behaviour, the app can crash with `NullReferenceException`. So, to overcome this issue, we will use the following code.

```
private void updateList()
        {
            try
            {
                int count = connectionsObj.TotalConnections.Count();
                // binding data to ListBox
                UserPanel.Visibility = System.Windows.Visibility.Visible;

                List<Information> _information = new List<Information>();
                for (int i = 0; i < connectionsObj.ConnectionsList.Count(); i++)
                {
                    string aFirstName = "";
                    string aLastName="";
                    string aHeadline="";
                    string anIndustry="";
                    string aPicUrl="";
                    string anId="";

                    if (connectionsObj.ConnectionsList[i].FirstName == null)
                    {
                        aFirstName = "";
                    }
                    else
                    {
                        aFirstName = connectionsObj.ConnectionsList[i].FirstName.Trim();
                    }


                    if (connectionsObj.ConnectionsList[i].Headline == null)
                    {
                        aHeadline = "";
                    }
                    else
                    {
                        aHeadline = connectionsObj.ConnectionsList[i].Headline.Trim();
                    }


                    if (connectionsObj.ConnectionsList[i].ID == null)
                    {
                        anId="";
                    }
                    else
                    {
                        anId = connectionsObj.ConnectionsList[i].ID;
                    }


                    if (connectionsObj.ConnectionsList[i].Industry == null)
                    {
                        anIndustry = "";
                    }
                    else
                    {
                        anIndustry= connectionsObj.ConnectionsList[i].Industry.Trim();
                    }


                    if (connectionsObj.ConnectionsList[i].LastName == null)
                    {
                        aLastName = "";
                    }
                    else
                    {
                       aLastName = connectionsObj.ConnectionsList[i].LastName.Trim();
                    }

                    if (connectionsObj.ConnectionsList[i].PictureUrl == null)
                    {
                        aPicUrl = "http://m3.licdn.com";
                    }
                    else
                    {
                        aPicUrl = connectionsObj.ConnectionsList[i].PictureUrl.Trim();
                    }

                    _information.Add(new Information(aPicUrl,aFirstName,aLastName,aHeadline,anIndustry));
                }

                connectionsList.ItemsSource = _information;
            }
            catch (Exception ex)
            { }
        }
```

We are just checking above that if any of the key has null value then it would be replaced by a valid value.

- The Information class is simply defined as shown below:

```
public class Information
{
    public string PictureUrl { get; set;}
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public string Headline { get; set; }
    public string Industry { get; set; }

    public Information(string _pictureurl, string _firstname, string _lastname, string _headline, string _industry)
    {
        this.PictureUrl  =_pictureurl;
        this.FirstName = _firstname.Trim();
        this.LastName = _lastname.Trim();
        this.Headline = _headline.Trim();
        this.Industry = _industry.Trim();
```


(C) Copyright Nokia 2013. All rights reserved.

```
    }
}
```

**Note:** To load images we've used **PhonePerformance.dll** as described in the Employees app with XML parsing and messaging in WP7 article.

## Build and Run

Now you may build the app and try to run it.

## References

- http://nuget.org/
- http://nuget.org/packages/Hammock
- http://developer.linkedin.com/documents/connections-api


(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.