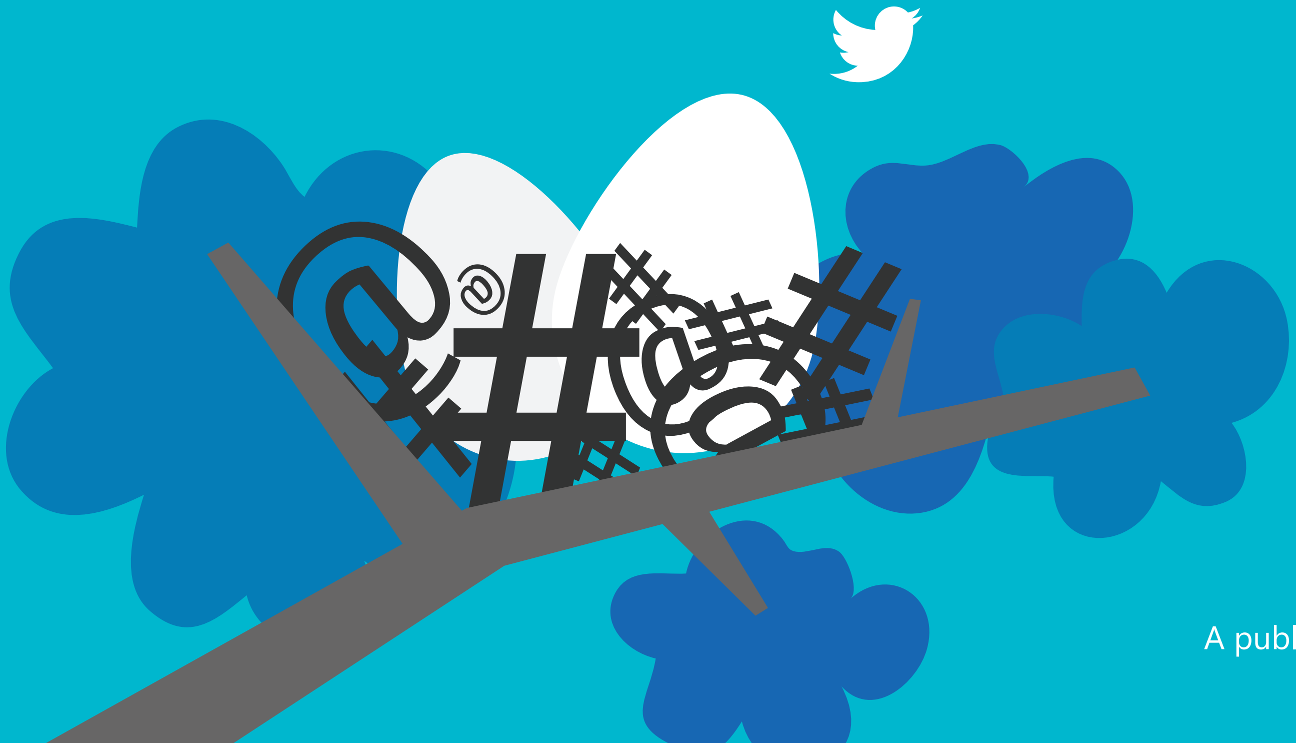
HOW TO CREATE A TWITTER APP FROM SCRATCH USING RADCONTROLS FOR WP8

By: Michael Crump



A publication of telerik

Table of Contents

ABOUT THE AUTHOR

Michael Crump

Michael Crump is a Microsoft MVP, INETA Community Champion, and an author of several .NET Framework eBooks.

He speaks at a variety of conferences and has written dozens of articles on .NET development. He works at Telerik with a focus on our XAML control suite.

You can follow him on Twitter at @mbcrump or keep up with his various blogs by visiting his Telerik Blog or his Personal Blog.

THE RIGHT TOOLS FOR THE RIGHT JOB

In order to create any type of professional application, you need the right tools. Here, we are going to write a Twitter app that allows the user to type in a twitter username and display the tweets from the user. We will be using the new Twitter API v1.1 released late last year and a couple of excellent tools to complete the project.

1. The Windows Phone 8 SDK needs to be installed manually if you already have a version of Visual Studio 2012 installed. You may use the web installer or download the .ISO file found here:

2. Telerik's RadControls for Windows Phone 8 contains over 60 controls and components to cut your development time.

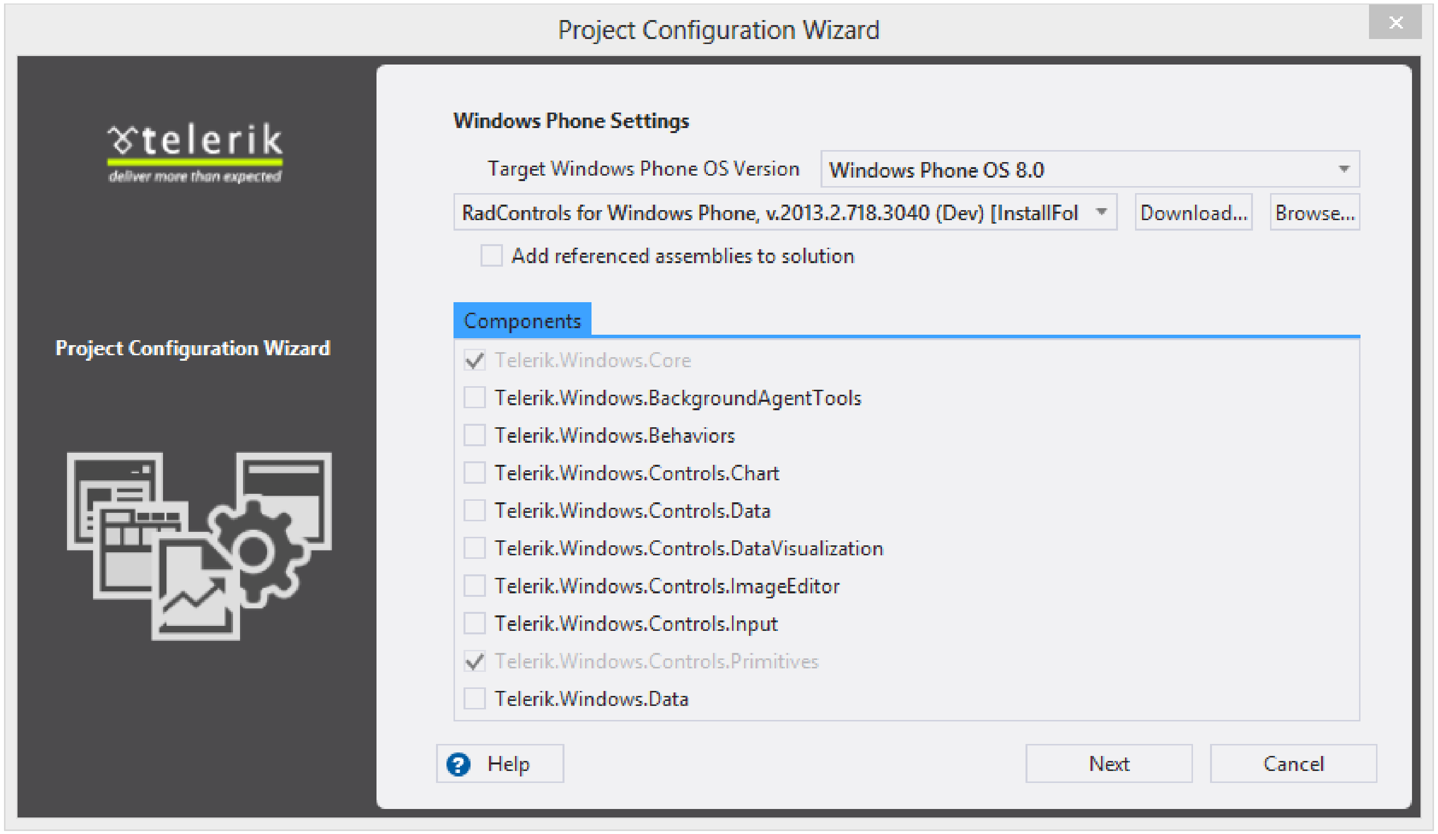
3. Finally, we need the ability to talk to Twitter's API endpoint. This is where Joe Mayo's Linq2Twitter library will come in.

After all of these have been installed, you are ready to proceed to the next part.

CONSTRUCTING OUR APP

GETTING SETUP

Launch Visual Studio 2012 and select Visual C# | Windows Phone | RadControls for Windows Phone and give it a meaningful name. On the project configuration wizard screen, leave the default component selected as shown in **Figure 1** and press next.



Project Configuration Wizard

Windows Phone Settings

Target Windows Phone OS Version: Windows Phone OS 8.0

RadControls for Windows Phone, v.2013.2.718.3040 (Dev) [InstallFol

☐ Add referenced assemblies to solution

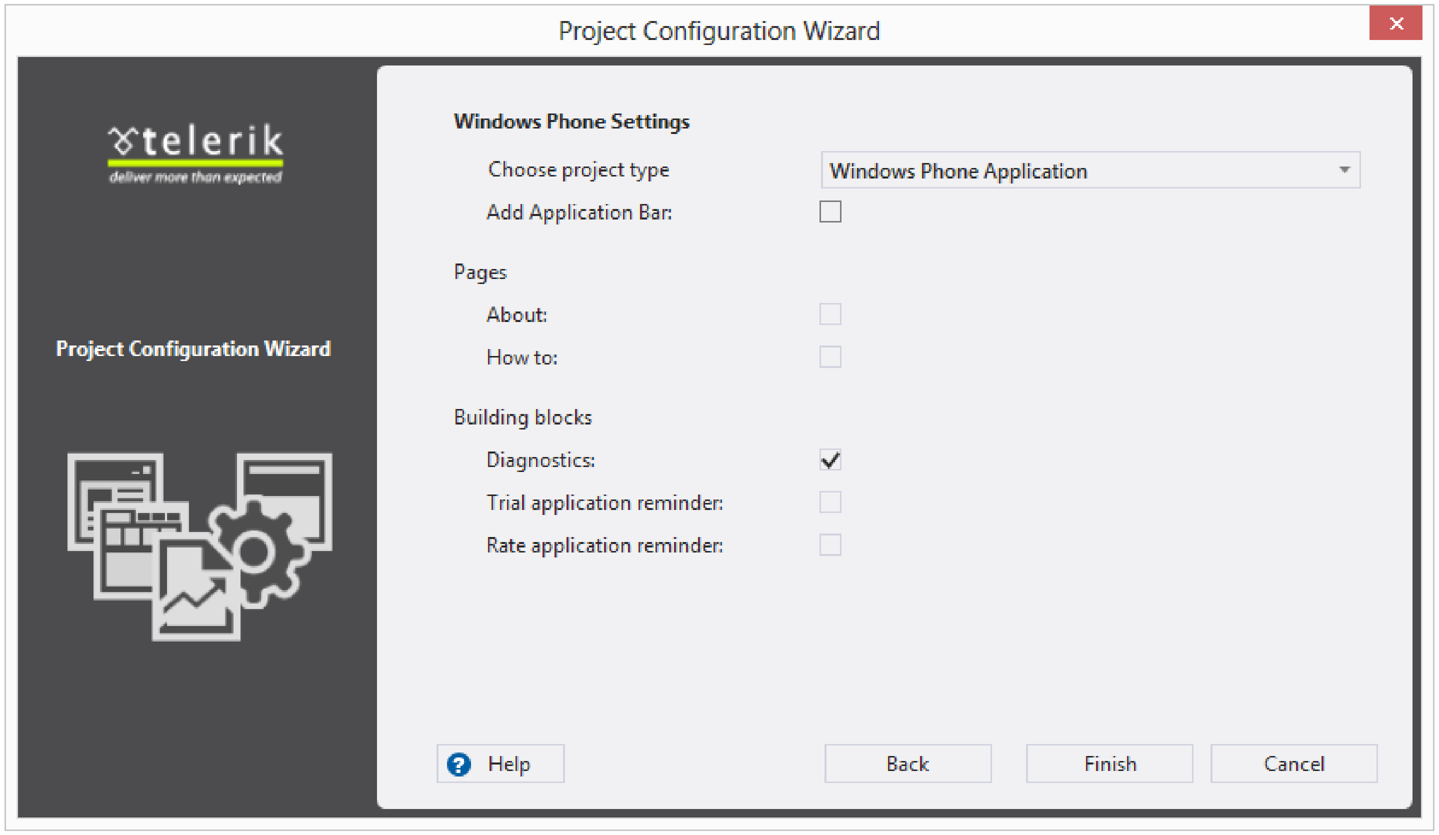
Components

- [✓] Telerik.Windows.Core
- [ ] Telerik.Windows.BackgroundAgentTools
- [ ] Telerik.Windows.Behaviors
- [ ] Telerik.Windows.Controls.Chart
- [ ] Telerik.Windows.Controls.Data
- [ ] Telerik.Windows.Controls.DataVisualization
- [ ] Telerik.Windows.Controls.ImageEditor
- [ ] Telerik.Windows.Controls.Input
- [✓] Telerik.Windows.Controls.Primitives
- [ ] Telerik.Windows.Data

Figure 1:

The First Screen to our RadControls for Windows Phone 8 App.

The second page in the project configuration wizard is shown in **Figure 2.**


Project Configuration Wizard

Windows Phone Settings

Choose project type: Windows Phone Application

Add Application Bar: ☐

Pages

About: ☐

How to: ☐

Building blocks

Diagnostics: ✓

Trial application reminder: ☐

Rate application reminder: ☐

Figure 2:

The Second Screen to our RadControls for Windows Phone 8 App.

As you can tell from **Figure 2**, not only can you change the default project type and add or remove an application bar, but we have built into the wizard two of the most common pages in a professional Windows Phone app.

- The **"About"** and **"How To"** page.

We have also added the ability to quickly add functionality such as error diagnostics, trial and rate reminders, which allows us to add common functionality that we typically code in every app with a single checkmark. Let's examine each of them briefly:

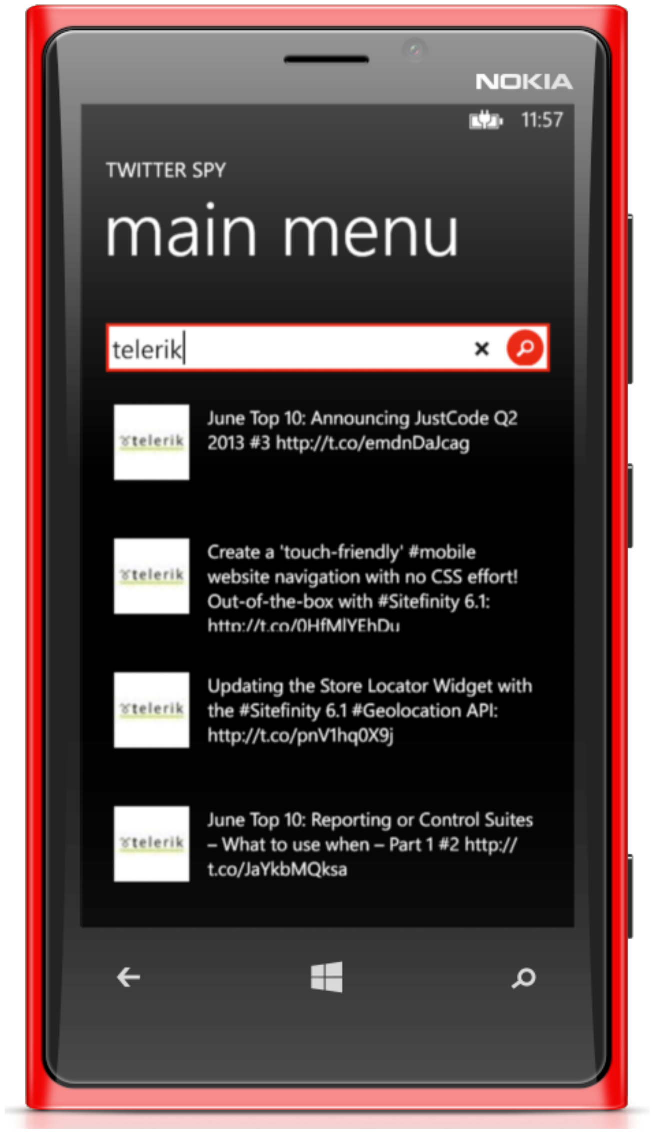
- Diagnostics: If your application crashes, this control allows you to collect the error information and email it to the author.

- Trial application reminder: Will remind your user that the application is a trial and prompt them to purchase it. You can specify when the reminder will display in a variety of ways.

- Rate application reminder: Will remind your user to rate the application in the store marketplace. There are also many different configuration options available. The error diagnostics will trap any unhandled exceptions and the trial and rate reminders will prompt your users to either purchase the app or rate the app depending on whether you desire this functionality or not.

We are building a simple first app, so remove all the checkmarks except Diagnostics.

Our UI will consist of RadTextBox and RadDataBoundListBox. Both of these controls contain the needed functionality to get started quickly. A screenshot of the final app is shown in **Figure 3**.



June Top 10: Announcing JustCode Q2 2013 #3 http://t.co/emdnDaJcag

Create a 'touch-friendly' #mobile website navigation with no CSS effort! Out-of-the-box with #Sitefinity 6.1: http://t.co/0HfMIYEhDu

Updating the Store Locator Widget with the #Sitefinity 6.1 #Geolocation API: http://t.co/pnV1hq0X9j

June Top 10: Reporting or Control Suites – What to use when – Part 1 #2 http://t.co/JaYkbMQksa

Figure 3: Our Final Windows Phone 8 App.

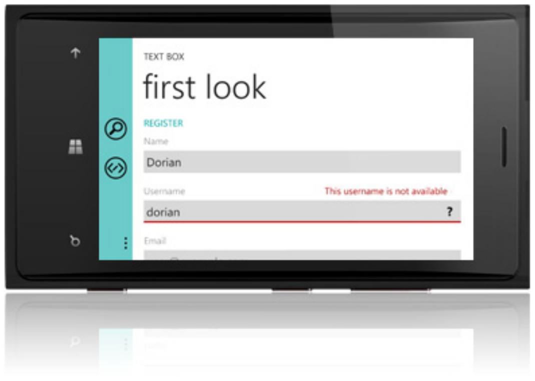
DEFINING OUR USER INTERFACE.

Now that we know how our final app will look, let's define our user interface in the ContentPanel Grid in MainPage.xaml:

```
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">

    <Grid.RowDefinitions>
        <RowDefinition Height="80*" />
        <RowDefinition Height="533*" />
    </Grid.RowDefinitions>

<telerikPrimitives:RadTextBox x:Name="txtUserName" Watermark="Enter Twitter Us
erName" ActionButtonVisibility="Visible" ActionButtonTap="txtUserName_ActionBu
ttonTap" >
        <telerikPrimitives:RadTextBox.ActionButtonStyle>
            <Style TargetType="telerikPrimitives:RadImageButton">
                <Setter Property="ButtonShape" Value="Ellipse"/>
            </Style>
        </telerikPrimitives:RadTextBox.ActionButtonStyle>
    </telerikPrimitives:RadTextBox>

<telerikPrimitives:RadDataBoundListBox Grid.Row="1" Name="lstTwitter" IsPullTo
RefreshEnabled="True" RefreshRequested="lstTwitter_RefreshRequested">
        <telerikPrimitives:RadDataBoundListBox.ItemTemplate>
            <DataTemplate>
                <StackPanel Orientation="Horizontal" Height="110" Margin="10">
                    <Image Source="{Binding ImageSource}" Height="73" Width="7
3" VerticalAlignment="Top" Margin="10,10,8,10"/>
                    <TextBlock Text="{Binding Message}" Margin="10" TextWrappi
ng="Wrap" FontSize="18" Width="320" />
                </StackPanel>
            </DataTemplate>
        </telerikPrimitives:RadDataBoundListBox.ItemTemplate>
    </telerikPrimitives:RadDataBoundListBox>

</Grid>
```

USING TWO CONTROLS TO CONSOLIDATE DEVELOPMENT TIME.

1. RadTextBox allows us to gather data from the end user. By using this control we can easily implement the following features without writing additional code or adding additional buttons:



Name: Dorian

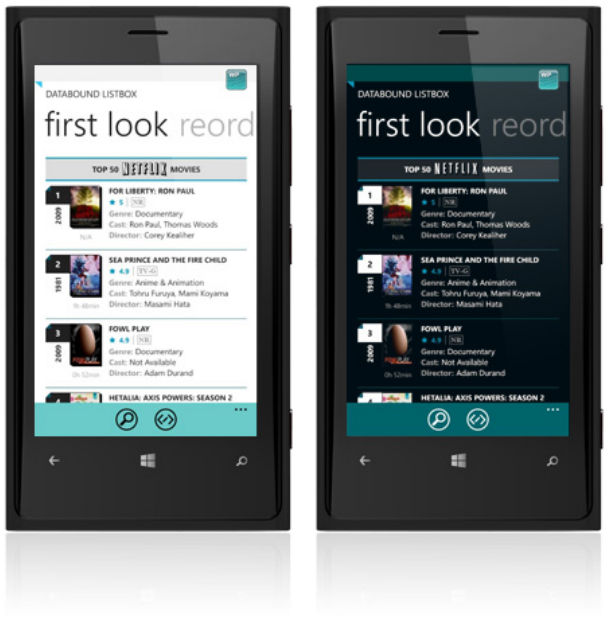
Username: dorian

This username is not available

Email

- We added a watermark so the user knows what data this field is expecting.
- There is a built in button that we can add by setting the ActionButtonVisibility to Visible and adding an event handler on ActionButtonTap.
- We can easily style the ActionButton to have an Ellipse shape as shown above.
- After the user starts typing, they can quickly clear the contents by pressing the X in the left hand corner, similar to Windows 8 Textbox fields. (This functionality occurs automatically.)

2. RadDataBoundListBox will allow our users to have a powerful control that handles many items as well as Pull-To-Refresh functionality. This allows the end-user to request a data refresh by pulling the top edge of the scrollable content down and releasing it. Inside of the ItemTemplate, we are going to create a DataTemplate that contains an Image and a TextBlock. The Image will show the user's twitter avatar and the TextBlock will contain the text of the tweet.



1 2009 FOR LIBERTY: RON PAUL ★ 5 | NR
Genre: Documentary
Cast: Ron Paul, Thomas Woods
Director: Corey Kealiher
N/A

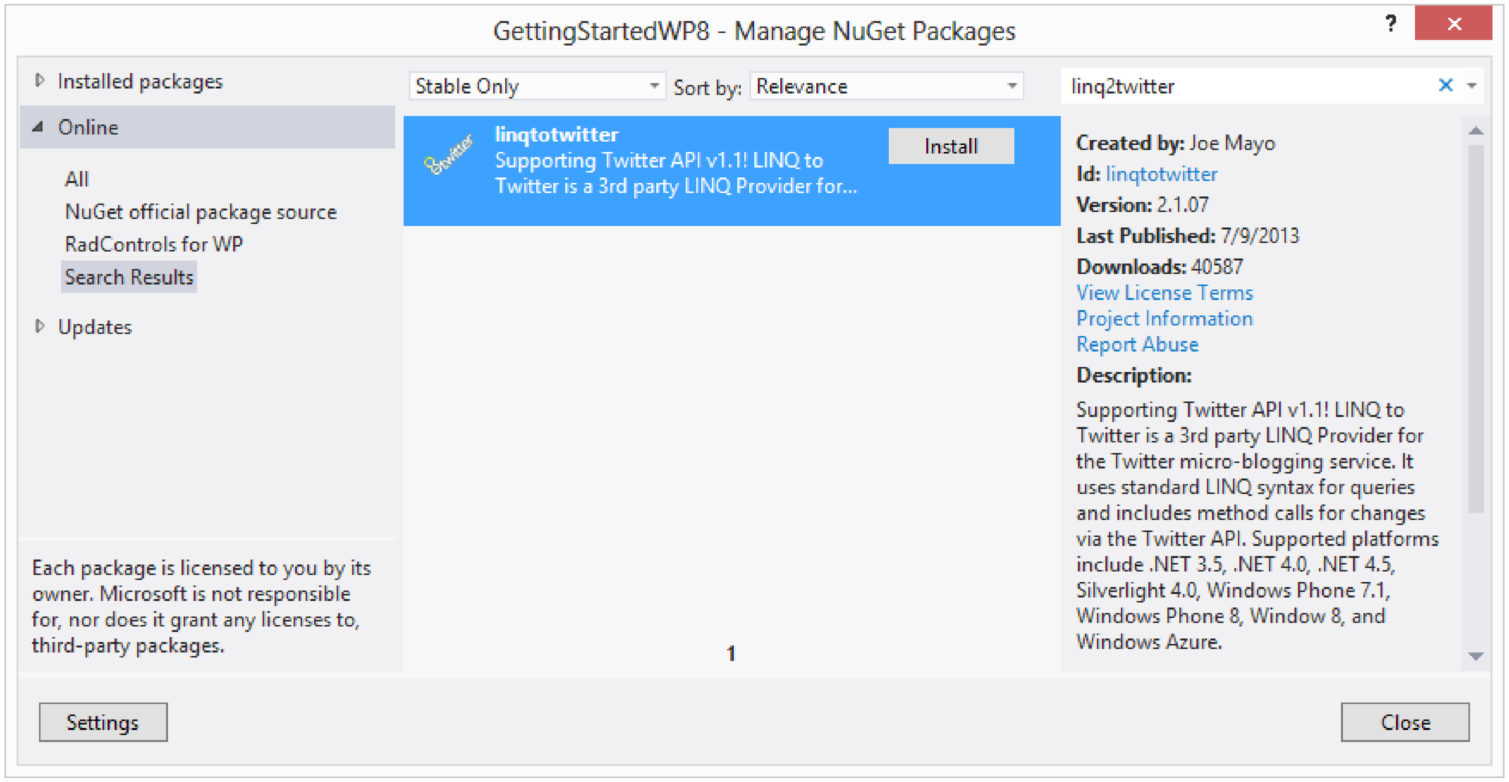
2 1981 SEA PRINCE AND THE FIRE CHILD ★ 4.9 | TV-G
Genre: Anime & Animation
Cast: Tohru Furuya, Mami Koyama
Director: Masami Hata
1h 48min

3 2009 FOWL PLAY ★ 4.9 | NR
Genre: Documentary
Cast: Not Available
Director: Adam Durand
0h 52min

HETALIA: AXIS POWERS: SEASON 2

EXAMINING THE TWITTER API 1.1

We will be using the Linq2Twitter library using NuGet. Right-click on references and select, **"Manage NuGet References"** then type in linq2twitter and click install as shown in **Figure 4**.


GettingStartedWP8 - Manage NuGet Packages

Installed packages
Online
All
NuGet official package source
RadControls for WP
Search Results
Updates

Each package is licensed to you by its owner. Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.

Stable Only | Sort by: Relevance | linq2twitter

linqtotwitter
Supporting Twitter API v1.1! LINQ to Twitter is a 3rd party LINQ Provider for...
Install

Created by: Joe Mayo
Id: linqtotwitter
Version: 2.1.07
Last Published: 7/9/2013
Downloads: 40587
View License Terms
Project Information
Report Abuse
Description:
Supporting Twitter API v1.1! LINQ to Twitter is a 3rd party LINQ Provider for the Twitter micro-blogging service. It uses standard LINQ syntax for queries and includes method calls for changes via the Twitter API. Supported platforms include .NET 3.5, .NET 4.0, .NET 4.5, Silverlight 4.0, Windows Phone 7.1, Windows Phone 8, Window 8, and Windows Azure.

1

Settings | Close

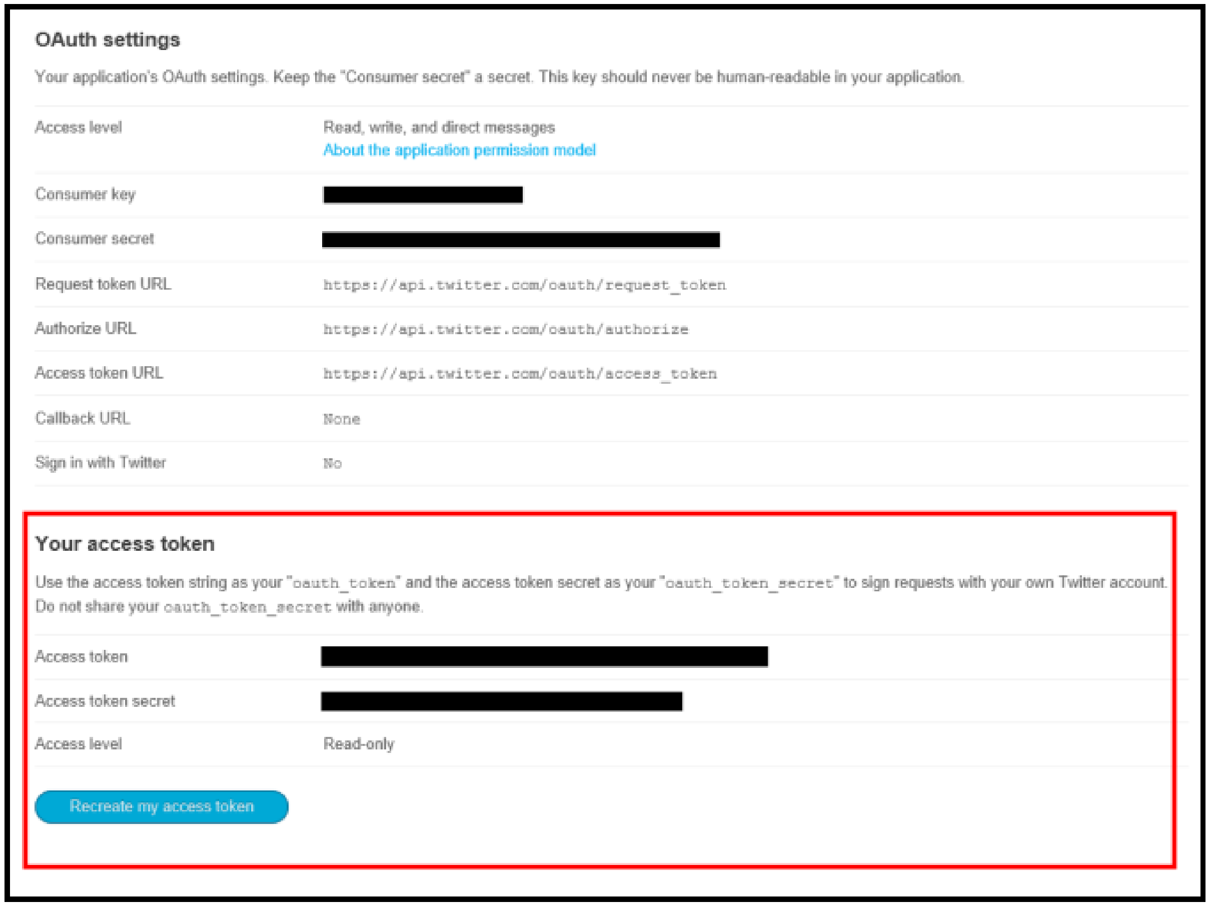
Figure 4: Adding Linq2Twitter to our Windows Phone 8 App.

Once installed, we can check **"References"** and should see **"LinqToTwitterWP"** added to our project.

WITH THE NEW TWITTER API COME NEW CHANGES

Twitter API 1.1 requires authentication on every API endpoint. That means that from now on you will need to create an app that contains your Consumer Key, Consumer Secret, Access Token and Access Token Secret. You can easily create an app by visiting https://dev.twitter.com. Once that is in place, you can get your keys by visiting your apps page.

Note that the developer will have to manually create their access tokens in the app's settings page as shown in **Figure 5**.



OAuth settings

Your application's OAuth settings. Keep the "Consumer secret" a secret. This key should never be human-readable in your application.

Access level	Read, write, and direct messages About the application permission model
Consumer key	[redacted]
Consumer secret	[redacted]
Request token URL	`https://api.twitter.com/oauth/request_token`
Authorize URL	`https://api.twitter.com/oauth/authorize`
Access token URL	`https://api.twitter.com/oauth/access_token`
Callback URL	None
Sign in with Twitter	No

Your access token

Use the access token string as your "`oauth_token`" and the access token secret as your "`oauth_token_secret`" to sign requests with your own Twitter account. Do not share your `oauth_token_secret` with anyone.

Access token	[redacted]
Access token secret	[redacted]
Access level	Read-only

Figure 5:

The OAuth Settings Page.

Begin by creating a simple class called TwitterItem and adding the following two properties. Make sure you mark the class as public as shown below:

```
public class TwitterItem
{
  public string ImageSource { get; set; }
  public string Message { get; set; }
}
```

Switch over to our MainPage.xaml.cs and before our MainPage constructor, we will need to add in the following code:

```
SingleUserAuthorizer singleUserAuthorizer = new SingleUserAuthorizer()
{
    Credentials = new SingleUserInMemoryCredentials()
    {
        ConsumerKey = "YOUR_CONSUMER_KEY",
        ConsumerSecret = "YOUR_CONSUMER_SECRET",
        TwitterAccessToken = "YOUR_ACCESS_TOKEN",
        TwitterAccessTokenSecret = "YOUR_ACCESS_TOKEN_SECRET"
    }
};
```

In this snippet, we are using Linq2Twitter to authenticate with Twitter about who we are and it will automatically determine what our permissions are. We can now drop in a method to Load Tweets once the user presses the search button (included in the RadTextBox) or refreshes the list with RadDataBoundListBox. The method is listed below:

```
public void LoadTweets()
{
    if (singleUserAuthorizer == null || !singleUserAuthorizer.IsAuthorized)
    {
        MessageBox.Show("Not Authorized!");
    }
    else
    {
        var twitterCtx = new TwitterContext(singleUserAuthorizer);
        (from tweet in twitterCtx.Status
            where tweet.Type == StatusType.User &&
                tweet.ScreenName == txtUserName.Text
            select tweet)
        .MaterializedAsyncCallback(asyncResponse =>
         Dispatcher.BeginInvoke(() =>
         {
                 if (asyncResponse.Status == TwitterErrorStatus.Success)
                 {
                           lstTwitter.ItemsSource =
                           (from Status tweet in asyncResponse.State
                               select new TwitterItem
                               {
                                   ImageSource = tweet.User.ProfileImageUrl,
                                   Message = tweet.Text
                               })
                           .ToList();
                 }
                 else
                 {
                          MessageBox.Show("Error: " + asyncResponse.Exception.
Message                              );
                 }
         }));

        lstTwitter.StopPullToRefreshLoading(true);
    }
}


private void txtUserName_ActionButtonTap(object sender, EventArgs e)
{
    LoadTweets();
}

private void lstTwitter_RefreshRequested(object sender, EventArgs e)
{
    LoadTweets();
}
```

We first check to see if we have authenticated properly and if we aren’t then inform the user. If we are authenticated then we are going to select the tweets with the username typed into our TextBox with an Async callback and add the results to our RadDataBoundListBox’s ItemSource property. Finally, we will turn off the Refresh loading animation from our RadDataBoundListBox.

FINISHING UP

UNDERSTANDING WHAT WE BUILT AND HOW WE DID IT

We saw just how quickly you could get up and running with a new Windows Phone 8 project by using a couple of controls and a great open-source library called Linq2Twitter. We were able to complete this project in a fraction of the time compared to using only standard Microsoft Controls and libraries.

ADDITIONAL RESOURCES

Before we go, I'd like to share several other resources that may help you with your next Windows Phone 8 application.

ToDoList – Shows how we built our ToDoList application as well as a series on creating the wireframes and building the actual app.

Picture Gallery – Demonstrates how to create a native Windows Phone client for Flickr or another image service.

How to promote your Windows Phone app on a budget whitepaper – Once your app is built, you will need to learn how to market it. This whitepaper offers tips and tricks from both the developer and marketing perspective. It is something you will definitely want to read.

Check out our Showcase Gallery on other apps built with RadControls.