# Feed Me Hash

This is a documentation that will go over the FeedMeHash project using the Twitter API Version 1.1. The project was created using C# .NET Version 3.5. This project was created by Anna Luong.

## Overview

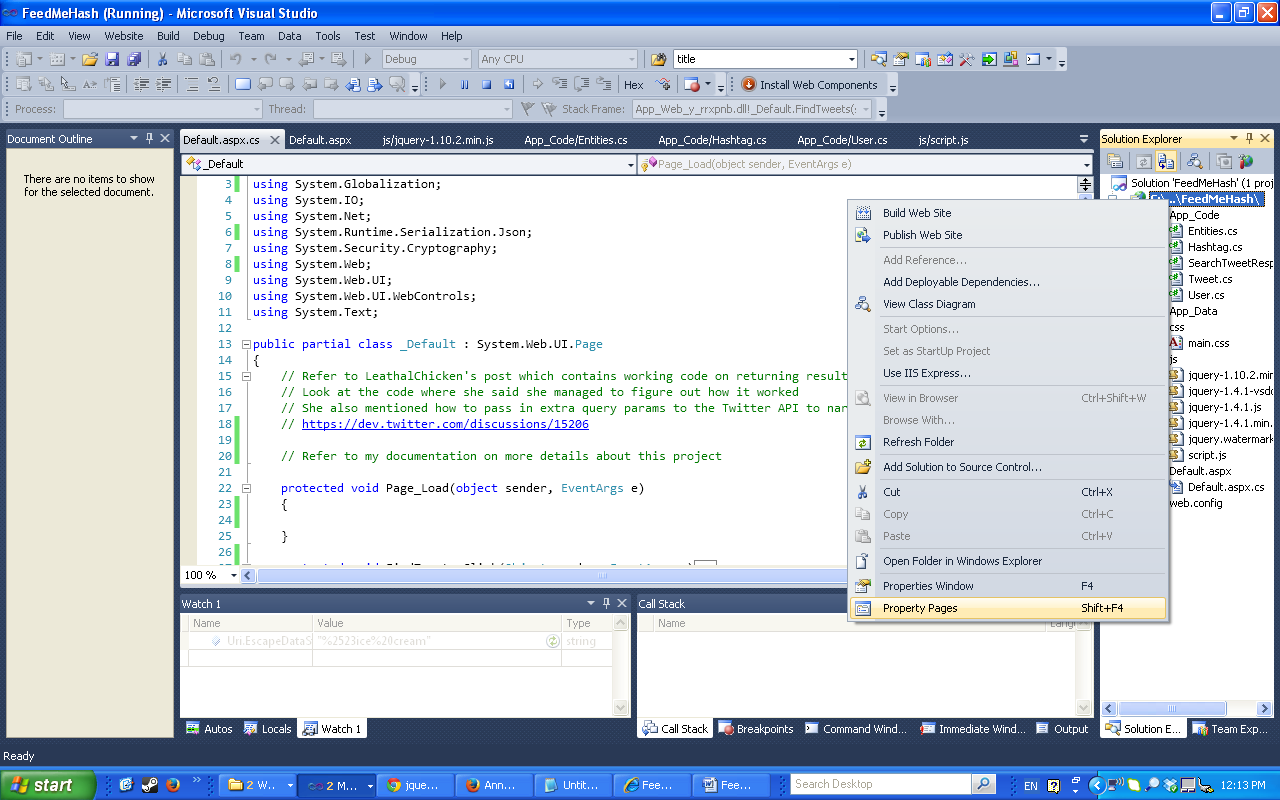
The document contains details of the usage of this project and its features. Details and features are as follows:

* Application Setup
* Find Tweets
* Filters
* Sort By
* Additional information

## Application Setup

If you haven’t done so, go to this URL (<https://dev.twitter.com/apps>) to create your application. Once you create your project, it is important to generate your access token for your project. Twitter API Version 1.1 requires authentication to make requests to the API.

When the website was first created, it was automatically created in .NET 2.0. To upgrade the website to version 3.5, right click the website and go to “**Property Pages**”.



On the left hand side, click on “**Build**” and select “**.NET Framework 3.5**” as the Target Framework. Apply and hit “**Ok**”.

You may want to right click “**Default.aspx**” and select “**Set As Start Page**”, to make sure this page always starts first when you run the website.

## Find Tweets

Refer to this discussion link (<https://dev.twitter.com/discussions/15206>) . This link contains a discussion where working code was found to create the authentication required by Twitter API Version 1.1. With this code, there is no need to use a third party program to get pass the authentication.

Note: If there is a need to add extra parameters in the URL. You would need to go to your twitter application. Go to the “OAuth Tool” and enter the URL and the query parameters you want to pass into the URL. OAuth Signing Results will be returned. The “Signature base string” **MUST** match exactly as was output:

Request URI: https://api.twitter.com/1.1/search/tweets.json

Request query: q=%23freebandnames&count=4

Signature base string:

|  |  |
| --- | --- |
|  | GET&https%3A%2F%2Fapi.twitter.com%2F1.1%2Fsearch%2Ftweets.json&**count%3D4**%26oauth\_consumer\_key%3DcSi9xDJd0SXDx0G2NzUXA%26oauth\_nonce%3Df145a0e66823dfe0ca2e591507030c6e%26oauth\_signature\_method%3DHMAC-SHA1%26oauth\_timestamp%3D1380387058%26oauth\_token%3D1909688060-o5PeClchGXZZhrlq9NvHsMCiCnhzEyX2vDVPbKG%26oauth\_version%3D1.0%26q%3D%2523freebandnames |

Default number of tweets returned is 15. Results will be returned via JSON string. DataContractJsonSerializer was used to deserialize the JSON string into C# objects. It also allows flexibility to create strongly typed properties not based on the dictionary key of the JSON string. It just needs to be mapped correctly, but the class property name can be given any name. Not all of the properties are mapped, just the ones that are needed.

Validation to check for an empty field and invalid input has been placed on the “**hashtag**” field using JQuery. An error message will display if validation is false. If a server error occurs while trying to find tweets using the Twitter API, a generic error message will display.

## Filters

This section will go over the functionality of the filters. Currently, this website will filter out the results by 1 hashtag entered in the form field. It can be changed to filter the results out by multiple hashtags, but it will kept simple for this project.

Since filters are done using PostBack, it will make a call to the FindTweet function and the results may be slightly different as it will pull the latest tweet, but it will still filter out the hashtag of the results from being displayed on the screen.

Validation to check for invalid input has been placed on the filter form field. It is ok if the field is empty.

## Sort

This section will go over the functionality of the sort feature. Currently the user has 2 options to choose from for sorting. Sort by Date (Ascending) or Sort By Date (Descending). Again, the features will be kept simple for this project. JQuery was used to sort the results.

Note: Since the Sort was done in JQuery, when a user sorts before entering a filter, the sort will be reset.

## Additional Information

This section will go over additional information for the reader about this project.

Simple CSS has been used to style the website just like how it was shown on the requirements document. No additional styles were added. If a design mock-up of the project was provided, the HTML/CSS will be done to make it look as close to the design as possible, but for the purpose of this project, it will just be a simple style.

The design on the requirements document did not contain a tweet date. The tweet date has been added to the results, so the user can visually see that the tweets are being sorted by date.

Watermark JS has been added to the project so that the user will be able to know what to enter in the form field.

The application consumer keys and access tokens are placed within the default page code behind. In the future, it could be placed in the Web.config file, but since this is a simple project, it’ll be placed in the code behind of the default.aspx page.