|  |  |
| --- | --- |
| Hi everyone. Today we’re going to take a look through some sample code I just cleaned up, which is an integration between Twitter and Excel. On my Twitter account, I have “liked” several hundred hilarious tweets, and I wanted to have them offline so I can read them again on long plane trips, or in case Twitter ever decides to remove that feature. Not sure if this sample code will be useful in your work, but it does highlight a few interesting things.  First, I used TweetSharp a NuGet package which is a .NET wrapper around the Twitter API. I found TweetSharp to be quite a robust and friendly wrapper to use.  Additionally, the code uses the interop services for Microsoft Excel to create a new Excel workbook and worksheet, enter data, format that data, and then to save the file to disk.  Finally, the code uses multithreading to ensure that the UI remains responsive, even when the heavy lifting of downloading tweets and saving them in Excel is ongoing. For the almost 800 favorite tweets in my account, the download takes about 15 seconds.  Let’s take a look through the code itself. | Run through the app live |
| When the form loads, it tries to read the Twitter credentials from a file called twitter-credentials.xml. This is a set of 4 pieces of information that come from your Twitter developer page. I’ve put it in an XML file so that it can be easily read from disk and not embedded in the code.  The code here simply tries to read the 4 pieces of information needed, and then confirms that each of the 4 variables has something reasonable at the end of the function. And of course we have exception handling in case the file is corrupt or missing. | Show form and double click to get form-load.  Then go to the LoadCredentials() function |
| Once we click Go, we set up a new BackgroundWorker thread, and set the function that we want to run in the background, the callback to be made upon completion, and the callback to be made when we want to report progress. This last bit is quite important, as it’s the way we’re going to indirectly update our UI from the worker thread, as updating the UI directly from a worker thread is generally not safe.  We then tell the worker thread to start, passing it the filename of the Excel workbook we want to create. | btnGo\_Click() |
| Here’s where the heavy lifting occurs. We get a reference to the worker thread so that we can post progress notifications later on. We set up our Excel writer class (which we’ll look at more closely in a minute), and the TweetSharp instance. The Twitter API won’t give us every tweet we’re looking for in one function call, so we’ll ask for at most 200 at a time.  You’ll notice that TweetSharp and my code use the term “favs” or “favorites” – this is because Twitter used to call “likes” “favorites”.  Once we have some number of favorite tweets, we simply iterate through each one, and send them to the Excel writer class instance, passing the timestamp of the tweet, it’s unique ID, some information about the author of the tweet, and the tweet itself.  Then we get set to tell Twitter which tweet to start from for the next time we ask for 200 favorite tweets, since we don’t want to get the same 200 we just asked for.  Finally, we report our progress back to the UI thread. With the way the Twitter API works, I don’t know how many liked tweets we have in total, so I can’t calculate percentage completed. But I can post back to the UI thread the number of tweets we’ve processed so far, which is somewhat helpful. | ExportTweets() |
| Alright, let’s take a look at the Excel portion of things. I’ve created a class to wrap the Excel API and to be a bridge between it and the Twitter side of things.  Here in the constructor, we create an instance of the Excel application, workbook, and worksheet, and then set up the column headers we want, and we want the column headers to be bold underline. | ExcelTweetWriter constructor |
| Then for each Tweet we want to write, we go to the next available row (which we’ve been keeping track of), and set the values. Note that we want to format some of the values in a specific way. We want the ID to be just a bunch of digits, not exponential format, which is the default. We also want the timestamp to be displayed unambiguously, in something that looks like ISO 8601 format. | ExcelTweetWriter - Write |
| When we’ve written all the Tweets to the file, the file hasn’t yet been written to disk, so we have to take care of that. First, we make sure the columns are wide enough to display all the data – that takes care of a personal pet peeve of mine, and enhances usability. Then we simply call the SaveAs method, passing the file name – most other parameters aren’t needed for our application. | ExcelTweetWriter - Close |
| You can download this code on my Github page. I’ve given a sample twitter-credentials.xml file, but I’ve scrambled my own personal data – you’ll have to use your own.  Thanks for watching – hopefully you find this sample application useful. | GitHub |