Checklist/Design Challenges

Highlights

- Notifications to the user if error occurs (e.g. incorrect username or password, password and confirm password fields that don't match when registering); don't fail silently
- Not allowing users to go back to the Login/Registration page while logged in, instead re-directing them to the tweets page as most websites would
- Not allowing users to access the tweet page if not logged in
- How I implemented editing/delete/retweet – how I know which tweet edit/delete/retweet using a hidden input element
    - https://github.com/6170-fa14/djoss_proj2/blob/master/views/users/index.ejs#L30

Help Wanted

- MVC separation – this is a concept that has always been somewhat confusing for me – how did I do in this regard?

Design Challenges/Decisions

My schema is as follows (see end of document for diagram):
User = {username: String, password: String, following: [String]}
Tweet = {originalAuthor: User, author: User, content: String}

With regards the user, the username and password fields are straightforward and reasonable in my opinion. I made the decision to have the user IDs a user is following embedded within the User object so that I could easily check that when rendering the users page and decide what tweets to display and not display. I chose not to have the following field be an array of User objects since that would include superfluous information (e.g. why would I need the password of a user I'm following).

With regards to the tweet, having the originalAuthor and author fields actually be User objects seemed more reasonable since a tweet is more closely tied to a user's account. The originalAuthor field was implemented for the purpose of retweeting. A tweet's originalAuthor stays the same, but the author can change so I can associate the retweet with the "new" author. The content field is self-explanatory.

Since the focus of this project (and class) is on building robust, functional web apps, I chose to spend a minimal amount of time on the UI, instead choosing to focus my time on making sure the different paths through the app worked.

In implementing the tweet functionality, it was easy enough to check for session attributes and thus not allow users to spoof other users (at least not easily). Security was not a concern here, but I did implement some very basic security measures e.g. make the session cookie secret. I only enabled edit/delete functionality on tweets for which the username matched the session username, so as to not even give users the option to

modify tweets that weren't theirs. This is what social networks do in real life; don't allow users to run amok with others' posts.

In terms of following users, I chose to display only the tweets of the users the current user is following. Then, the user can increase the amount of content he/she is exposed to by following more people. Otherwise, the follow feature would be somewhat useless. I didn't implement unfollowing due to limited time, but it would be analogous to following.

In terms of retweeting, I chose not to allow users to retweet their own tweets. This is based on my limited understanding of Twitter (I've never heard of anyone retweeting his own tweets). I allowed users to retweet a tweet that has already been retweeted one or more times since I didn't see a downside to doing so, and users could follow the progression of retweets from one person to another.