Presence - Part 3

Due Date

This assignment is due by **11:59 PM on Tuesday, May 12.**

Assignment

Last week, we continued making progress on our Presence application, adding table views and fetching live data with web services. However, you may have noticed that because we were performing our Twitter API requests and image downloads on the main thread, there were stalls and hangs in the user interface. Additionally, working on a read-only Twitter client is only fun for so long- we want an application that allows us to update our own online "presence" as well.

This week, we'll employ multithreading in our application to avoid those hangs, and we'll use text input in a modally presented view controller to update our own status.

Here are the requirements for Part 3:

1. When you make a Twitter API requests, do it without blocking the main UI thread. You can **either use NSThread directly or the more convenient NSOperation and NSOperationQueue** classes. While you're fetching the timeline data, indicate loading progress in your user interface. Make sure to **call back to the main thread to update your UI** when complete. Remember, UIKit views must be accessed on the main thread!

2. Last week, you added code to fetch an image for each user by calling NSData and UIImage methods on the main thread. Like the Twitter API request, we don't want this to hang our app. **Create a subclass of NSOperation that downloads a single image**. Then, **manage the image downloads with an NSOperationQueue** in your person list controller. The queue should not allow more than one image download at a time. You'll need some mechanism for passing the loaded image back to the main thread and triggering its display.

3. It's time to update our own Twitter status using Presence as well. **Display a button on the right side of the navigation bar** when the person list controller is visible (you may need to refer back to a previous lecture!). When this button is pressed, **present a new view controller modally**. Let's call it the StatusComposeViewController. This view controller should **allow the user to type a status update, then either send it or cancel**. We're giving you some code, +[TwitterHelper updateStatus:forUsername:withPassword:], that takes care of sending a status update to Twitter. Again, make sure that when this method is called, the **main thread is not blocked** and the UI shows progress. **Dismiss the status compose controller after sending or canceling.**

There is an **accompanying archive titled Presence3Files.zip** which includes an updated TwitterHelper class. You can use the original TwitterUsers property list and JSON parsing code from Presence 2.

Testing

In most assignments testing of the resulting application is the primary objective. In this case, testing/grading will be done both on the behavior of the application, and also on the code.

We will be looking at the following:

1. Your project should **build without errors or warnings** and **run without crashing**.
2. Each view controller should be the File's Owner of its own Interface Builder document.
3. Remember the rules of retain, release and autorelease. **Don't leak memory or over-release**.
4. Readability is important. Make sure to **decompose, comment and name thoughtfully**.
5. Your program should behave as described above. It must fetch Twitter data, download images and update your status without stalling the user interface. The status update view controller should be presented and dismissed modally.

Hints and Reminders

Using NSThread directly
If you choose to use NSThread for loading Twitter user timelines, make sure to create & release your own autorelease pool in the detached thread method! If you use NSOperation with an NSOperationQueue, you don't need to do this.

StatusComposeViewController
Your status compose view should include two buttons, one for canceling and one for sending the status update to Twitter. You can either do this by creating UIButton instances and adding them to your view, or by **customizing your status compose view controller's navigation item and then pushing it onto a navigation controller**. Remember, a navigation controller is just another view controller, so you can present and dismiss it modally. If you take this route, your status compose view will look something like this:

Showing the keyboard
Remember, there is no API for showing or hiding the keyboard. It'll appear automatically when a text container becomes first responder. Pick reasonable text input traits for your text container.

Presenting and dismissing
From within your person list controller, in the action method for your navigation button, you'll want to instantiate a status compose controller and then present it. It'll look something like this:

```
StatusComposeViewController *statusComposeController = ...;
[self presentModalViewController:statusComposeController animated:YES];
```

Refer to the slides at the end of Lecture 11 for tips on dismissing the controller when your user is done. It's generally considered poor form for a modally presented view controller to dismiss itself. Instead, **the same object that presented it should dismiss it**. So, you'll need a loosely coupled way for the compose view controller to tell the person list controller that it's ready to be dismissed.

Twitter username and password
You'll need to specify a username and password when updating your status via **+[TwitterHelper updateStatus:forUsername:withPassword:]**. You can use your real Twitter account (if you have one), or just create one for testing.

You should hard-code the username and password somewhere in your code (unless you pursue the extra credit!). Make sure that we can **easily configure a username and password for testing** with your app. See the Flickr demo from lecture for an example of this.

Either way, **remove your username and password from your project before submitting**. You wouldn't want us posting from your account, would you?

Extra Credit
Here are some suggestions for extra credit:

- Instead of reading the TwitterUsers property list, fetch the data for your own friends! You'll need to modify the TwitterHelper class to add a new method for doing this. You can follow the example of the other methods in the file.
- Customize the table view cells in the person list to display each person's most recent status update inline. You may want to create a custom subclass of UITableViewCell to make it easier to dequeue and reuse cells.
- Twitter status updates must be 140 characters or less. Display to your user how many more characters they can type in a status update. If the user tries to type any more than the limit, prevent them from doing so using a text field delegate method.
- Run the clang tool on your project as discussed in lecture. Include the output with your submission, mentioning whether it helped you find any issues or encountered false positives.

If you undertake any extra credit, please let us know in your submission notes or otherwise. If you don't, we might not know to look for it. And be sure that the core functionality of your application is solid before working on any of this!