Presence - Part 2

Due Date

This assignment is due by **11:59 PM, May 5.**

Assignment

Last week, you created the first version of Presence, an iPhone application for viewing online status updates. This week, we'll improve the application in two major ways. First, we'll make the switch from hardcoded views to table views, so that our application can display large, dynamic data sets. Second, we're going to read local data from property lists and fetch remote data using web services, displaying actual content in our application.

Here are the requirements for Part 2:

1. **Read the contents of TwitterUsers.plist at an appropriate place in your application's life cycle**. You'll need to add the file to your Xcode project so that it's built into the application bundle. Use the contents to create a list of people to display. Handle common error conditions gracefully.
2. **Define a Person model object.** For now, a person should include an image, a username, a display name and a list of status messages. So, for each username in the property list above, you'll need to instantiate a person object and fetch their data from Twitter. The walkthrough covers this in more depth.
3. Update the PersonListViewController class to **manage a plain style UITableView**. You may want to make it a subclass of UITableViewController. **Display the list of Person objects in the table view**, including an image and a name.
4. **When a person is selected in the list**, set up and push a PersonDetailViewController
5. The PersonDetailViewController class needs to be updated as well. It should have a person property which a client can set. It will **manage a grouped style UITableView** where it **displays a list of status messages for the selected user**.

There is an archive accompanying this assignment titled **Presence2Files.zip** which includes the TwitterUsers.plist file that your application will read. Additionally, it includes a class called TwitterHelper which encapsulates requesting a user's status updates. Finally, it includes some code for parsing JSON. You won't need to call this code directly for now, but you will need to add it to your project.

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. **Do not put your entire application into a single Interface Builder document!**
3. You should be using retain, release and autorelease correctly at this point. **Don't leak memory or over-release**.
4. Since the project is getting more complex, **readability is important**. Add comments as appropriate, use descriptive variable and method names, and decompose your code.
5. Your program should behave as described above, reading in a list of Twitter users from the property list, fetching their info and status updates using the Twitter API, listing users in a table view and displaying their status updates when selected.

Walkthrough

Reading the TwitterUsers.plist file
You may first want to inspect the contents of the property list in a text editor. Once you're familiar with the structure, you'll want to use the NSBundle class to get the path for the property list. Many of the Foundation classes have constructors that let you specify a path- in this case, you'll probably want to use +[NSArray arrayWithContentsOfFile:].

The Person model object
It's pretty clear at this point that **we need a model object to package together all the bits of data relating to a person.** At the bare minimum, it should keep track of an image (or image URL), the username for Twitter API queries, a display name and a list of status updates. Keep in mind that **some users may not have a display name**.

Making Twitter API requests
The accompanying **Presence2Files.zip archive includes a class called TwitterHelper.** Here is the class definition:

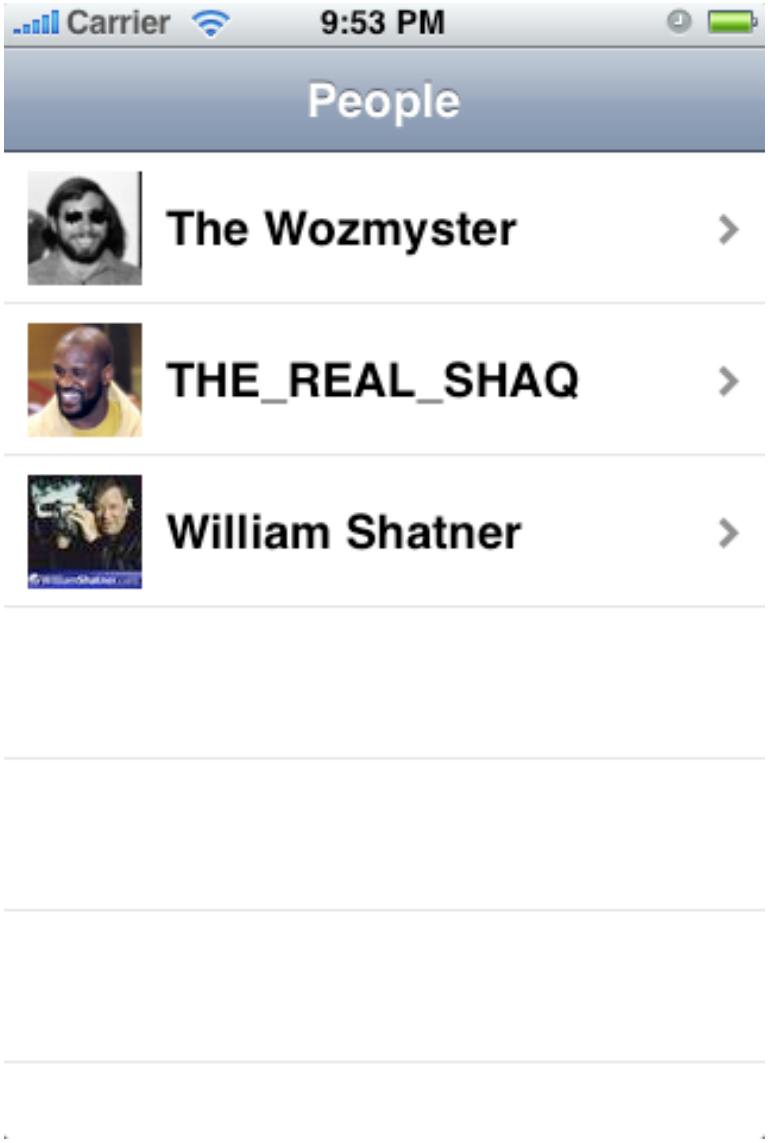
```
@interface TwitterHelper : NSObject {
}

// Returns a dictionary with info about the given username.
+ (NSDictionary *)fetchInfoForUsername:(NSString *)username;

// Returns an array of status updates for the given username.
+ (NSArray *)fetchTimelineForUsername:(NSString *)username;
```

You can **use the +[TwitterHelper fetchInfoForUsername:] method to get a display name, image URL and other info about a username.** You can then **use the +[TwitterHelper fetchTimelineForUsername:] method to get a list of status updates for a given user**. As noted above, this method will block until it succeeds or fails. This doesn't scale well for making multiple requests, nor does it do well in cases where network connectivity is slow or unreliable. Next week, we'll look at some ways of avoiding this.

Drag the header and source files for the TwitterHelper class into your Xcode project. Do the same with the folder titled JSON. For now, you don't need to know anything about what these classes are doing under the hood.

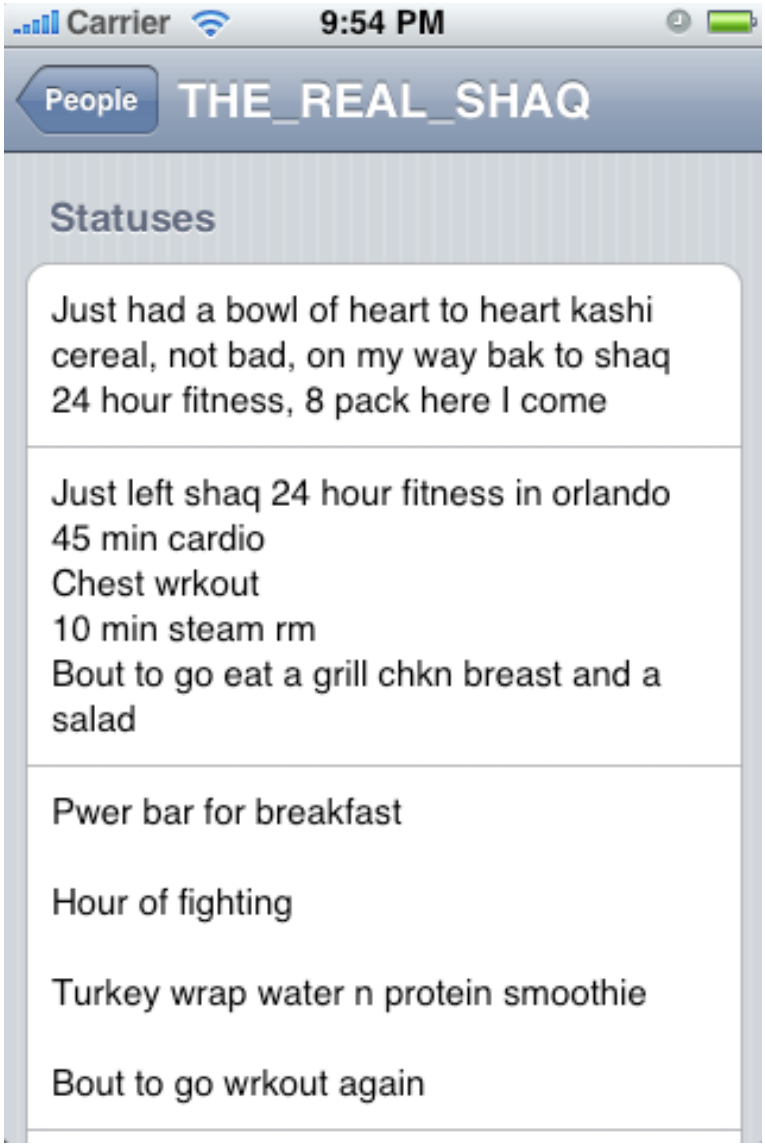
Optional: If you're curious, you can begin to check out the Twitter API. It's documented at http://apiwiki.twitter.com. After making Twitter API calls, the helper class receives responses in JSON format. To parse these, it utilizes the excellent json-framework. You can read more about it, check out sources and download the latest release at http://code.google.com/p/json-framework.

Managing a table view with PersonListViewController
You will probably want to use UITableViewController as the starting point for your view controller subclass. It automatically creates a table view with itself as the delegate and datasource, among other things. You don't even need to use a NIB if the table view is all you're displaying- just instantiate your subclass using -initWithStyle:. If you do choose to use a NIB and -initWithNibName:bundle:, be certain that your view outlet is pointing at a valid UITableView.



Responding to a selection
In Presence 1, we pushed a PersonDetailViewController onto the navigation stack in our button's action method. Now, we're going to do it in response to a table selection. You remember how to do that, right? Your PersonDetailViewController class should now have a "person" property. Be sure to set it on the instance after creation, before you push it onto the stack.

Displaying status updates in a table view
The big challenge here is the fact that status messages require varying amounts of vertical space. Luckily, there's a UITableView delegate method for specifying the height of each row. To compute the height of your text, **use the methods from UIStringDrawing.h**. In a table view with the grouped style, the cell is inset by 10 pixels from the edges, and the text is inset by an additional 10 pixels. UITableViewYou'll also need to customize the contentView of your table view cell by adding your own subview (refer to Lecture 8), since UITableViewCell doesn't handle multiline text by default.



Extra Credit

Here are some suggestions for enhancing the second version of Presence.

- Show the selected person's image in the detail view. Perhaps add a header view to the table?
- Add support for reordering and deleting users from the table view. Use the view controller's standard Edit/Done button as described in Lecture 7, and implement the required datasource & delegate methods for allowing rows to be reordered and deleted.
- Customize the status table view cell even further- display a timestamp for the status update, or anything else you like.
- Many Twitter status updates include a URL. Enhance your detail table view so that tapping a cell with a URL opens the URL in Safari. Accompany this with a visual hint of some sort in the cells that have a URL.

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!