Paparazzi - Part 4

Due Date

This assignment is due by **11:59 PM, February 26.**

Assignment

By this point, your Paparazzi app looks up real users on Flickr, finds and lists their photos and lets you view them up close by zooming and panning. It also plots these photos on a map and lets you browser photos by location. Not bad! You would almost be ready to post this on the App Store...

But before we are done with Paparazzi, you need to make your app feel a little zippier, and tie your Flickr users to the System's Address Book.

Here are the requirements for Part 4:

1. **Thread your URL requests**. In Paparazzi 3, you were likely requesting data from Flickr on your main (UI) thread, blocking user interaction while a fetch was in progress. Use NSThreads or NSOperations to get make your URL requests on a different thread and process the results in the background. Remember: UIKit views must be accessed on the main thread!
2. **Integrate with Address Book.** Add a blue "Detail" disclosure indicator to your table view cells in Contacts. When clicked, create and push an appropriate Address Book view controller for that user.
3. **Add your own whizzy feature for Paparazzi.** Integrate something new into your app to. WebKit? Photo Browser? Multitouch? You decide. It should make sense in the context of the Paparazzi app, but you are encouraged to think as far outside the box as you want. Just be sure to tell us about it in your ReadmMe.

There are no additional files included with Paparazzi 4. You should build on your app from Paparazzi 3.

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. It may be difficult to prove that your threading is working properly if your internet connection is fast. To test this, you can **turn on the "TEST_HIGH_NETWORK_LATENCY" flag in FlickrFetcher.h**. Turning this on will introduce an artificial delay to your UI if you are not threading properly. Your TAs *will* run this test for your project, so you should too.
6. For lists of people, display a blue disclosure button which pushes an Address Book view controller onto the navigation stack.

Walkthrough

Using NSThread directly

If you choose to use NSThread for loading Flickr requests, 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.

CoreData & Threading

CoreData is a powerful database access framework, but there are some gotchas if you don't use it right. When accessing CoreData objects on multiple threads, you need to use a different NSManagedObjectContext on each thread. For the purposes of this assignment, don't worry about solving this problem, and instead just send the results of your threaded request results back to the main thread. Have your main thread interact with the FlickrFetcher's shared ManagedObjectContext as you already do in Paparazzi 2 and Paparazzi 3.

Displaying the Blue Disclosure Button

You can set the accessoryView for your table cells when you create them. There is a special delegate method that is called when the disclosure button is pressed. See the UITableView documentation or UITableView.h

Address Book UI

Address Book View Controllers require an ABRecordRef representing a person in the Address Book to be passed in. When deciding what record to use when the disclosure button is pressed, consider that you may want to ask your user to fill out details for a new person in the Address Book.

You have a couple of options for associating your CoreData object with your Address Book record. You could either store the Flickr user name in the address book, or alternatively, store the AddressBook record ID in your database. Consider the advantages and disadvantages to each approach and choose which works better for you. Tell us about your approach in your Read Me.

Your whizzy feature

This is up to you. You don't need to do anything too fancy here, but pick a technology you want to try, and find an interesting way to integrate it into your app. Maybe you can launch to a Flickr URL in a WebView. Maybe you want to try rotating an image using Multitouch. Or perhaps you can just overlay some metadata information about a photo on your PhotoViewController.

Extra Credit

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!