Assignment VII: Multithreaded Places

Objective

We continue working on our Places application this week. The primary goal this time is to make our application perform well by offloading blocking activity (like network access) into separate threads. We'll also add a new feature: thumbnail images.

Be sure to check out the Hints section below!

Also, check out the latest in the Evaluation section to make sure you understand what you are going to be evaluated on with this assignment.

Materials

- You will still need your Flickr API key.

Required Tasks

1. Add a thumbnail image of each photo to every row in any table view that shows a list of photos. Do not spend network bandwidth fetching a thumbnail image until a photo's row appears on screen.
2. Use Grand Central Dispatch to move all network activity out of the main thread of your application so that it is never "blocked" from user interaction because it is waiting on the network.
3. Since they are small, thumbnail images should have their data cached in your Core Data database for Favorites and Recents.
4. You do not need to cache thumbnail images in the list of photos in a Top Place (but you do need to display them).
5. If the user is waiting for a full image of a photo (i.e. not a thumbnail) to download from the network at any point, show a spinning activity indicator in place of the image (i.e., don't just show a blank screen). Again, the photo should be downloading in a separate thread, so if the user gets tired of waiting he or she can just click the Back button to get out of it.
6. Add alphabetical sections to your Favorites list of places.

Hints

1. You will have two different implementations for displaying thumbnail images because you have two different kinds of table views that display photos. Your Core Data table views will automatically update if you modify a property (like the thumbnail image data) but your non-Core Data table view will not. You might find the `UITableView` method `reloadRowsAtIndexPaths:withRowAnimation:` helpful for the latter table view.
2. `CoreDataTableViewController`'s `thumbnailImageForManagedObject:` will likely be something you will want to override.
3. Because the required tasks say that you should not download Flickr data for a thumbnail until it is needed on screen, you should probably have a method in your `Photo` class which, when asked for a thumbnail image that has not yet been downloaded, returns `nil`, then queues up a block to download the data and set it in Core Data. Any table that requested the thumbnail (and got `nil` in return) will now get updated automatically when the block is done (because an attribute, the thumbnail data, will have changed).
4. Do **not** operate on an `NSManagedObjectContext` from more than one thread!
5. You will probably want `FlickrFetcherPhotoFormatSquare` for your thumbnails.
6. `NSFileManager` is thread-safe (as long as you don't use the same instance of one in two different threads) and so is `NSData`'s `writeToFile:`, so you can use these outside of your main thread.
7. Do **not** call any user-interface methods from a secondary thread (only from the main thread).
8. There is a "spinning wheel" view available for you in the UIKit. It's called `UIActivityIndicatorView`. You can drag one out into a `.xib` file or create it in code (be sure to use the proper initializer if so). Check the documentation (or click the right buttons in the Inspector in Interface Builder) for how to stop it, start it, and make it hide itself when it's not spinning. Be careful not to accidentally add it as a **subview** of your `UIScrollView` when you drag it out in Interface Builder. It wants to be a **sibling** of the `UIScrollView` in the view hierarchy not a subview (make sure it's in front of the `UIScrollView` though).
9. This is your last assignment. By now you should be comfortable tracking down bugs with the debugger and tracking down how to use unfamiliar API using the documentation in Xcode. Being able to do both of these things is just as important a part of being a solid iOS developer as knowing how to implement a particular feature. Don't cheat yourself of this last opportunity to get experience with these tasks.

Evaluation

In all of the assignments this quarter, writing quality code that builds without warnings or errors, and then testing the resulting application and iterating until it functions properly is the goal.

Here are the most common reasons assignments are marked down:

- Project does not build.
- Project does not build without warnings.
- One or more items in the Required Tasks section was not satisfied.
- A fundamental concept was not understood.
- Code is sloppy and hard to read (e.g. indentation is not consistent, etc.).
- Assignment was turned in late (you get 3 late days per quarter, so use them wisely).
- Code is too lightly or too heavily commented.
- Code crashes.
- Code leaks memory!
- User-interface hangs or is blocked from action by the user.

Extra Credit

If you do any Extra Credit items, don't forget to note what you did in your submission `README`.

1. Make your application work on the iPad with appropriate user-interface idioms on that platform (i.e. on the iPad, you do not have to use a `UINavigationController` for all three view controllers).
2. When you are caching your images in the file system, perhaps enforce some limit as to the total disk space you will allow the cache to consume.
3. Add searching to one or more table views.
4. Make your sectionization of your Favorite places efficient by storing the section name for a given place in a database attribute rather than implementing it as a method in your `NSManagedObject` subclass.