Assignment V: Places

Objective

In this series of assignments, you will create an application that presents a list of popular Flickr photo spots and let users pick favorites. This first assignment is to create a navigation-based application to let users browse the most popular spots from which photos uploaded to Flickr in the last day or so were taken, and then click to view photos from the places that interest them.

All the data you need will be downloaded from Flickr.com using Flickr's API.

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

- This is a completely new application, so you will not need anything (but the knowledge you gained) from your first four homework assignments.
- You will need to obtain a Flickr API key. A free Flickr account is just fine (you won't be posting photos, just querying them).

Required Tasks

1. Use the provided `FlickrFetcher` class method `topPlaces` to get a list of the most popular Flickr photo spots in the last day or so.
2. Create a `UITabBarController`-based user-interface with two tabs. The first shows a `UITableView` with the list of places obtained in Required Task #1. The second is a list of recently-viewed photos.
3. Anywhere a place appears in a table view in your application, the most detailed part of the location (e.g. the city name) should be the `title` of the table view's cell and any remaining description of the location (e.g. state, province, country, etc.) should appear as the `subtitle` of the table view cell.
4. When the user chooses a place from the list of places, you must query Flickr (using `FlickrFetcher`) to get recent photos from that place.
5. Any list of photos should display the photo's title as the table view cell's `title` and its description as the table view cell's `subtitle`. If the photo has no title, use its description as the title. If it has no title or description, use "Unknown" as the title.
6. When the user chooses a photo from any list, display it inside a scrolling view that allows the user to pan and zoom (a reasonable amount).
7. When a photo appears for the first time on screen, it should initially be zoomed to show as much of the photo as possible with no extra, unused space. It is not necessary to continue to do this as the user zooms in and out on the photo by pinching.
8. On the iPhone, all navigation from list to list or from list to photo display should be done inside a `UINavigationController`. Your entire application must work in both portrait and landscape orientations. Support for the iPad is optional.
9. The recents tab must show a list of the most recently view photos in chronological order of viewing with the most recent at the top, and no duplicates in the list. When a recent photo in the list is chosen, it should navigate to a view of the photo in exactly the same way as if the photo were chosen from the list of photos taken in a given place. Limit the size of the list of recents to a reasonable number.
10. The list of recents should be saved in `NSUserDefaults`.
11. Make the network activity indicator in the upper left corner of your application spin whenever you are accessing the network. You turn this on or off using the property `networkActivityIndicatorVisible` in `[UIApplication sharedApplication]`.

Hints

1. A good place to copy the helper files (`FlickrFetcher.[mh]`, `FlickrAPIKey.h` and the `JSON` directory) into your project in Xcode is "Other Sources."
2. Put your own Flickr API key into `FlickrAPIKey.h` or your queries will not work.
3. The very first thing you're probably going to want to do once you have copied the `FlickrFetcher` code into your application (and set your API key) is to do a `topPlaces` query and then `NSLog()` the results. That way you can see the format of the fetched Flickr results (it's an `NSArray` of `NSDictionary` objects). Ditto when you query Flickr for the list of photos at a given place.
4. The full name of a place is found using the `_content` key in a place's dictionary of info. A unique identifier for a place in its dictionary of info is `place_id`. The `place_id` of a place may not be the same as the `place_id` of a photo found by querying that place (because the photo's place might be street level and the place querying functions in Flickr only allow locality-level searching).
5. If you look carefully, you'll notice that the value for the key `description` in a dictionary of photo information from Flickr is not actually the photo's description. Instead, it's another `NSDictionary` that has a key `_content` in it. That's where the actual description is.
6. The key `id` (in a photo's dictionary of info) is a unique, persistent photo identifier.
7. When you create a new `UIViewController` using New File ... from the File menu in Xcode, be sure to check the box `UITableViewController` if the view your `UIViewController` is going to control is a table view. You probably don't need the "With .xib for user interface" button to be clicked because your view is probably just going to be the table view and nothing else (in fact, none of your `UIViewController`s in this application probably needs a `.xib` file). `UITableViewController` implements `loadView` for you, by the way.
8. The various MVC's in this application are very similar. Be sure to use good object-oriented programming design techniques to share as much code between them as makes sense and still have each of them have a well-defined, standalone API.
9. There are plenty of ways to sort your list of places, but believe it or not, it can be done with a single line of code. Check out the method `sortedArrayUsingDescriptors:`, it's pretty powerful when you want to sort an array of dictionaries.
10. Sometimes Flickr returns 0 places when you query `topPlaces`. It might be that it only allows a certain number of queries for a given Flickr API key in a given amount of time. In any case, if you wait a bit, it'll start working again. Another option is to cache the results of this call (for debugging purposes only!) into `NSUserDefaults`. Then you won't be banging on Flickr so much as you iterate on your code. The code

you submit should be making live Flickr queries, not using a cache. If this hint is confusing you, ignore it.

11. Do not communicate data between view controllers on your navigation stack using global data! Nor should an MVC that is pushed onto a navigation stack have a pointer back to the MVC that pushed it (unless it is a blind pointer via delegation, but that shouldn't be necessary for this assignment). Each MVC should be set up with the information it needs before it is pushed and then allowed to go do its thing. It is more crucial than ever that you understand which MVC's you need and what the Model of each one is!

12. The method `mutableCopy` in `NSDictionary` and/or `NSArray` might come in handy when you want to add something to a data structure (like your Recents) that is already stored (immutably) in `NSUserDefaults`. The mutable copy that is returned is owned by the caller (i.e. it violates the rule that only methods that start with `alloc/copy/new` return objects the caller owns--this method acts like `copy` in that respect). The mutable copy you get back is only "shallowly" mutable (e.g. making a mutable copy of an array of arrays will only return the top-level array mutable, the arrays inside it will still be immutable if that's what they were before the mutable copy).

13. It is recommended to use the Flickr photo size `Large`. This is usually 1024x768, which will fit nicely to the resolution on the iPad and also on iPhone 4's. It's a bit of overkill for pre-iPhone 4 hardware, but that's okay. See the Flickr doc about sizes if you want to learn more.

14. Required Task #7 (initial photo zooming) requires some calculations involving the size of the `UIScrollView`'s bounds and the size of the photo. Remember from lecture where geometry calculations for a view have to occur.

15. You're going to notice that your application is not very responsive! Whenever it goes off to query Flickr, there'll be a big pause. This is very bad, but we will be learning how to solve this next week, so don't waste your time trying to fix it now.

16. Here are some fun methods in the SDK which may (or may not) come in handy for this assignment. This is just a hint, so you are not required in any way to use these methods.

    a. `componentsSeparatedByString:` & `componentsJoinedByString:`

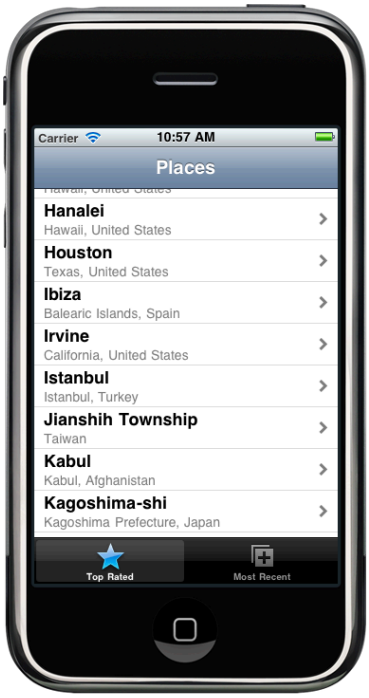
    b. `sortedArrayUsingDescriptors:` & `sortDescriptorWithKey:ascending:`

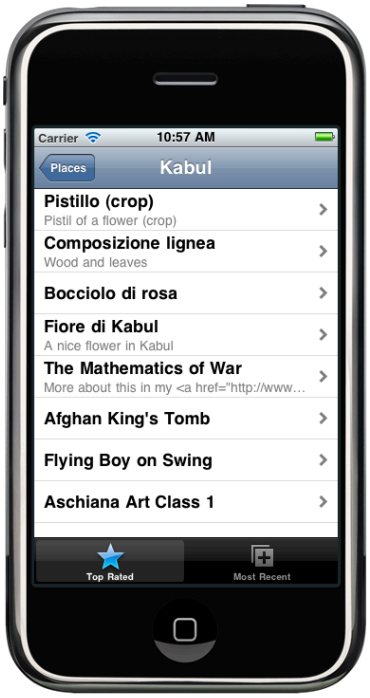
    c. `deselectRowAtIndexPath:animated:` & `indexPathForSelectedRow`

    d. `filterUsingPredicate:` & `predicateWithFormat:`

Screen Shots

These screen shots are only meant to be examples. Your actual implementation may vary from this as long as required tasks are completed. Also, the data from Flickr may be changing and might be different from what is displayed below.





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!

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.
2. Let users delete items from the Recents list. Swiping is sort of a "you have to know about it" way for the user to delete so you'll probably want to add an Edit button on the Recents page to put the table view into editing mode.
3. Divide your places list into sections (each section a different letter of the alphabet). You can then implement `sectionIndexTitlesForTableView:` to get a nice index into your sections to appear on the right-hand side of the table.
4. Divide your photos list into sections too where each section is how may hours ago the photo was taken (the list you get back should already be approximately sorted into that order, so you should not have to sort to do this). If you implement EC#3 above in a reusable, object-oriented fashion, this extra credit can be implemented in as few as 3 lines of code. But if you did not, it could be dozens! You'll want to know about the Flickr key `dateupload` (which is the number of seconds since 1970 that the photo was uploaded). The `NSDate` method `timeIntervalSince1970` might also prove helpful.