IvanCaldwell / **ios-weather-objc**
forked from LambdaSchool/ios-weather-objc

No description, website, or topics provided. Edit

Manage topics

5 commits | 1 branch | 0 releases | 1 contributor

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

This branch is even with LambdaSchool:master.

SpencerCurtis Update README.md

WeatherObjC	Add starter project with assets and example images	
.gitignore	Add .gitignore	
Example1.jpg	Add starter project with assets and example images	5 months ago
Example2.jpg	Add starter project with assets and example images	5 months ago
README.md	Update README.md	5 months ago

Clone with HTTPS Use SSH
Use Git or checkout with SVN using the web URL.
https://github.com/IvanCaldwell/ios-weat
Open in Desktop | Download ZIP

README.md

Weather Obj-C

Introduction

The goal of this project is to use the OpenWeatherMap API to create an app that will display a 7 day forecast for any zip code the user enters.

Instructions

Please fork and clone this repository. Use the provided Xcode project as it contains some icon images in the Assets folder. Commit regularly as you complete the requirements in this project.

Part 1 - Storyboard

1. In a `UIViewController` scene, add a `UISearchBar` and a `UICollectionView`.
2. In the collection view cell, add an image view and a label. The image view will show an icon of the weather for the forecast, and the labels will show the temperature and the city respectively.
3. Create Cocoa Touch Subclasses for the custom cell and your view controller scene. Set up any outlets and actions necessary.

Part 2 - Model/Model Controller

1. Go to this signup page and sign up for a new OpenWeatherMap account. Once you create your account (and you may have to sign in if it doesn't automatically sign you in), go to the "API keys" section of your "My Home" page. Generate a new API key.
    - Refer to this page of the API's documentation. Even though it says 16 day forecast, you will only get a 7 day forecast as your account's API key is free. Look the example JSON here to familiarize yourself with the structure of the dictionary that will be deserialized using `NSJSONSerialization`.
2. Create a model object that represents a daily forecast. It should include:

- The name of the city (yes, this is slightly repetitive)
- The temperature
- A `UIImage` that will be the icon for the forecast (sunny, rainy, etc.)

3. Create a normal memberwise initializer.
4. Create an initializer that takes in a dictionary and the city name. (Refer to the example JSON again, and/or ask a PM if you're unclear on why you should pass the city name in separately)
5. Create a model controller, and add a property of an array of forecasts.
6. In the initializer of the model controller, give an initial value of an empty array for the forecasts property.
7. Add static constants to the model controller for the base url and your API key.
8. Create a function that will perform a data task to get the forecasts. This should take in a zip code, and have a completion block with an error. The process should be the same as if you were writing this in Swift up to the point where you would use a `JSONDecoder`. Instead, use `NSJSONSerialization` to turn the data returned from the data task into a dictionary. From there, you will be able to parse it to the point where you can pass a sub-dictionary to your model's initializer.

Part 3 - View Controller

1. In your view controller's .h file, adopt the `UISearchBarDelegate` and `UICollectionViewDataSource` protocols. Also add a property for the model controller.
2. In the view controller's initializers (`initWithCoder` and `initWithNibName`), initialize the model controller property.
3. In the `viewDidLoad` set the view controller as the `delegate` of the search bar and the `dataSource` of the collection view.
4. Implement the required `UICollectionViewDataSource` methods.
5. Implement the `searchBarSearchButtonClicked` method to trigger your fetch forecast method from the model controller. Reload the table view on the main queue in the completion block of the fetch forecast method call.

Go Further

1. Add additional properties to the model such as the day of the forecast, highs and lows for the day, wind speed, etc. Update the model's initializers to properly parse the dictionary and initalize the new propeties.
2. Change your collection view cell to look more like the common forecast you would see on the news. There are a couple example images included in the repo to help you get some ideas. Feel free to be creative with this.