Turn To Tech

Project Name - NavCtrl

Assignment
*links to reading material at end of document
ARC must be disabled for all NavController project

Assignment 1
Initial Setup

1. Create a git repository.
2. Add all the existing files of this project to the repo and commit them.
3. As you make changes that have been tested you should add & commit these changes to git.
4. Tag your commits after each assignment below is completed.

Assignment 2
Add Basic Features

1. Create a branch for assignment 2
2. Add two more companies to the start page
3. Add 3 products for each company
4. Show the logo for each company and each product (similar to the UITableView project)
5. Add a 3rd level to the app so that a web page related to the product is shown when a product is selected using the two approaches below:
    a. Use the class UIWebView
    b. Change the UIWebView to WKWebView and learn about the differences between the two (pros and cons). Note: You will have to do this programmatically
6. Enable delete functionality so that companies and products can be deleted.
    a. Make sure that when you leave the view and come back that the product is still deleted
    b. The deleted product should only come back if you restart the app
7. Allow the user to rearrange the order of the cells
8. Commit and then push your work to github
9. Make sure you can recreate this project on a different computer using github.

Assignment 3
Refactor your project to use an Object Oriented Approach with DAO

1. Create a branch for this assignment
2. Create a Company class and a Product class
3. Refactor the project to be object-oriented, so that each company is a Company object and each product is a Product object.
4. The Company class should have a products property (an array of Product objects).

5. Add a Data Access Object (DAO).
    a. The DAO is a separate class where all the Companies and Products are created.
    b. Make sure you are clear about the MVC design pattern.
6. Refactor the app so that the companies and products are actually created in the DAO, and NOT in the view controller.
7. Ensure that there can only ever be one instance of the DAO class. In other words, it should follow the Singleton design pattern. You should be able to explain the benefits of this pattern.
8. Commit and then push your work to github as you implement various features. Make sure whatever you commit is in working condition
9. Once the assignment is complete, tag this branch with a label e.g. “assignment3-V1”. Understand the purpose of tags.
10. Make sure you can recreate this project on a different computer using github.

Assignment 4
Adding and Editing

1. Continuing on the branch for assignment3 above:
    a. Enable ‘add’ functionality so that users can add new companies and products and define their properties with a form.
    b. Enable ‘edit’ functionality so that users can update the properties of existing companies and products with a form.

Assignment 5
More Network Tools

1. Continuing on the branch for assignment3 above:
    a. Use an online API to get stock quotes for each company and display the current stock price next to the company name. You should only make one request to get data for all the companies. This should update whenever the company view is displayed. Make the request asynchronously using NSURLSession.
    b. Add and commit to git

Assignment 6
NSUserDefaults

1. Read and understand the “About the iOS File System” section of this document:
    a. https://developer.apple.com/library/ios/documentation/FileManagement/Conceptual/FileSystemProgrammingGuide/FileSystemOverview/FileSystemOverview.html

2. Zip up your project at this time so you can always go back.
3. From the Assignment 3 branch create a new git branch called “NSUserDefaults”
4. Change the persistence type to Object Archiving with NSUserDefaults.

a. (***ask for the NSUserDefaults sample project at this time***)
    i. When your app loads it should check to see if data is saved in NSUserDefaults. If it's not there, the app should create the data using hard coded values and save it to NSUserDefaults. If it is there, it should only read the data to create objects and not use the hard coded values.
    ii. If all above steps are done correctly, this should not require any changes to the View Controllers
    i. If you delete a product and restart the app, the product should still be deleted

11. Save the objects to a file, instead of NSUserDefaults. The API call is very similar to what you used for creating NSData from your objects.
    a. Locate the file on the file system. What is the format of the file?

    b. Use the command line tool **plutil,** convert the above file to a readable version. View that file in a text editor. Do you recognize the data from the objects you persisted?

Assignment 7
SQLite

1. Switch back to the Assignment 3 branch and then create a new branch for this assignment
2. If you are not familiar with SQL go to the SQL list on https://trello.com/b/U1vTHuDd/learning-resources and do some tutorials. Understanding how relational databases work is fundamental to being a strong software developer. Recommended: Khan Academy, sqlzoo.
3. Make sure you've got SQLite Browser installed: http://sqlitebrowser.org/
4. Use SQLite Browser to create a database with two tables (Company & Product) and populate them with all of your data.
5. Add an id field to the Company table and add a company_id field to the Product table. Assign each product its corresponding company id. You will use this later to match up each product with its appropriate company programmatically.
6. Save this database to your Desktop.
7. Drag and drop the file into your project (make sure that "copy if needed" is selected)
    a. In Build Phases make sure you can find it in the "Copy Bundle Resources".
8. Add persistence to NavCtrl using sqlite (***ask for the SQLite sample project at this time***)
        i. When your app loads it should check to see if the database exists in the Documents directory of your simulator. If it's not there, the app should copy it from the bundle to the documents directory.

ii. If all above steps are done correctly, this should not require any changes to the View Controllers
iii. If you delete a product and restart the app, the product should still be deleted

9. Zip up your project at this time so you can always go back.

Assignment 8

Memory Management

1. Using XCode's 'Instruments' tool, ensure that there are no objects allocated that should not be there, and that there are no leaks
2. Zip up your project at this time so you can always go back.

Assignment 9

Core Data

1. Create a new branch off of SQLite called ‘core-data’
2. Remove all code relating to Object Archiving with NSUserDefaults and/or SQLite and replace your method of persistence with Core Data (*ask for the Core Data sample project at this time*)
    a. When your app loads it should check to see if data is saved in Core Data. If it’s not there, the app should create the data using hard coded values and save it to Core Data. If it is there, it should only read the data to create objects and not use the hard coded values.
    b. The NSManaged Object for Company should have a relationship in Core Data with the NSManagaed Object for Product.
    c. If all above steps are done correctly, this should not require any changes to the View Controllers. This means that the View Controllers should not interact with NSManagedObjects—only NSObjects
    d. If you delete a product and restart the app, the product should still be deleted

Assignment 10

UICollectionView

1. Create a new branch that splits off of core-data called “UICollectionView”
2. Refactor the app so that instead of using UITableView, it uses UICollectionView
    a. If the above steps are done correctly, this should not require any changes to the DAO or Models
    b. User should be able to delete from collection view
    c. OPTIONAL: add the option to rearrange the collectionView cells

Assignment 11

AFNetworking

1. Refactor the app to use AFNetworking instead of NSURLSession

Completion Checklist

- App should work with or without internet. Stock quotes should start showing once network is back without restarting
- Stock quotes should refresh every 60s
- The DAO header file should be identical for all three persistence options above
- In the Core Data version, View Controllers should not interact with NSManagedObjects directly
- Understand forward class declaration and why we need it.
- Profile your app using Instruments and make sure you have only the required objects in memory
- Understand how Allocations and Mark Generations works
- Reading
    - **Archives and Serializations Programming Guide** -- read till page 20 https://developer.apple.com/library/ios/documentation/Cocoa/Conceptual/Archiving/Archiving.pdf
    - **Core Data Programming Guide** https://developer.apple.com/library/mac/documentation/DeveloperTools/Conceptual/InstrumentsUserGuide/InstrumentsUserGuide.pdf First 80 pages only
    - **Instruments User Guide.**
        - Make sure you read the introductory chapter and **Locating Memory Issues in Your App**

Reading

1. View Controller Programming Guide for iOS
    a. https://developer.apple.com/library/ios/featuredarticles/ViewControllerPGforiPhoneOS/ViewControllerPGforiOS.pdf

2. View Programming Guide for iOS
    a. https://developer.apple.com/library/ios/documentation/WindowsViews/Conceptual/ViewPG_iPhoneOS/ViewPG_iPhoneOS.pdf

3. Core data Programming Guide
    a. https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/CoreData/CoreData.pdf

4. Advanced Memory management programming guide
    a. https://developer.apple.com/library/ios/documentation/Cocoa/Conceptual/MemoryMgmt/MemoryMgmt.pdf