Saving Data in iOS

Hands-On Challenges

Saving Data in iOS Hands-On Challenges

Introduction to FMDB

While SQLite provides everything you need to working with an embeddable database in your application, using its C API can be a little clunky to the Objective-C developer. Thankfully, you have the Flying Meat Database, better known as FMDB, to assist you with such a task.

FMDB is simply an Objective-C wrapper around SQLite, providing an objected orientated interface for working with the database. It also adds the additional behavior of querying columns by name, versus index, which makes for cleaner code.

This challenge returns you to the old time radio app. Before starting this challenge, it is recommended that you try the previous challenge. That way, you can get an idea of how the app works but also, how it works with SQLite. In this challenge, you will be converting those SQLite calls to FMDB, giving you a hands on demonstations of the time saving features of FMDB.

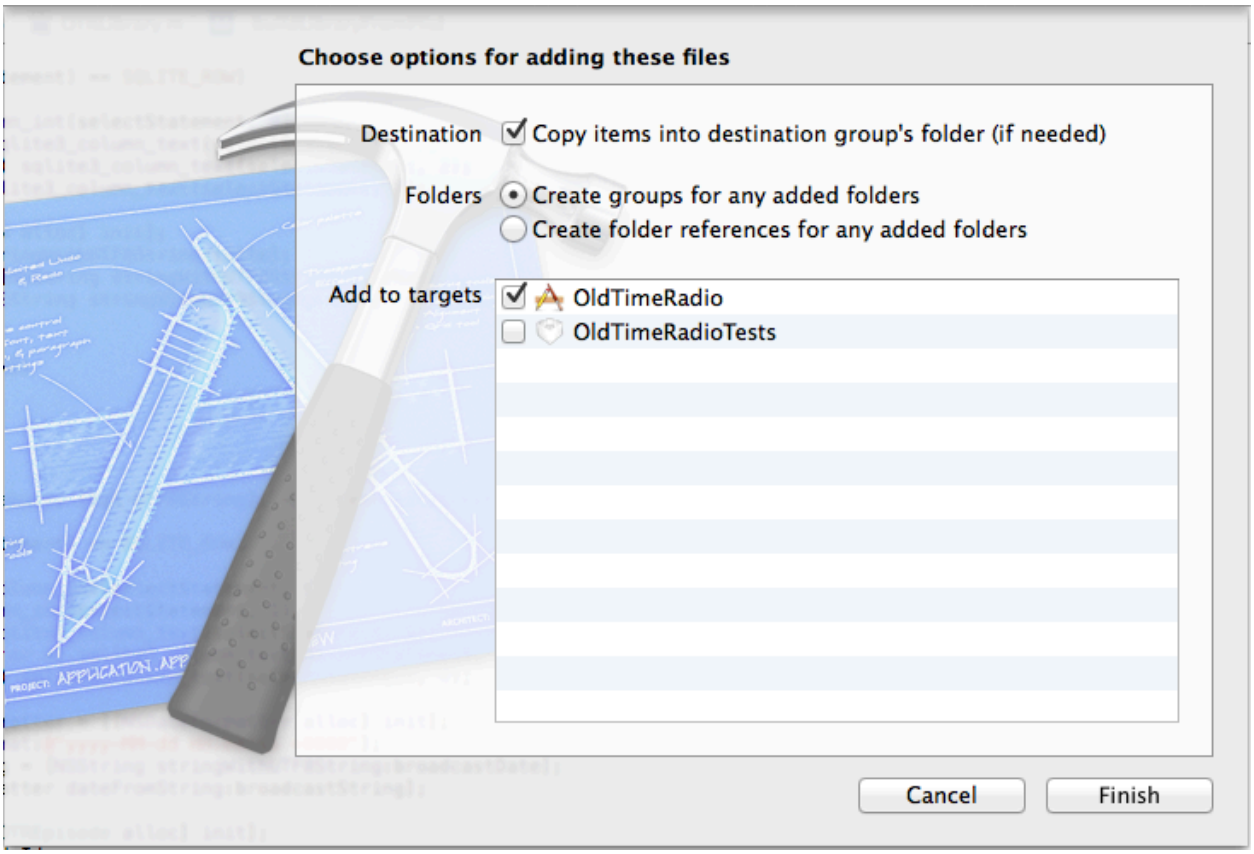
Challenge

Open the starter project and open **OTRLibrary.m**. Just a quick glance will show a multitude of SQLite API calls. Your job is convert these calls to FMDB. But first, you need to install FMDB proper.

Open your web browser of choice and navigate to this link:

https://github.com/ccgus/fmdb

Download the project, uncompress it, and import it to your project. Open the uncompressed folider in your finder, and navigate to the **src** directory. To import, drag the **extra** and **fmdb** folders into your project. You will see the following dialog:



Once you have those folders imported, you are now ready to start using FMDB.

First, create a singleton object to access the database from your application. From the file menu, **select File / New / File.** Choose **Objective-C class** and name it, **FMDBManager**.

Open the **FMDBManager.h** file and add the following import startement at the top of the file:

```
#import "FMDB.h"
```

Just underneath the interface declaration, next add the following property:

```
@property (readonly, nonatomic) FMDatabase * database;
```

Once the object has been created, gaining access to the database is as easy as accessing a property. Note that it is a readonly property. This is so the object can handle the actual setup and teardown of the object.

Now, add the following three methods:

```
+ (id) sharedManager;
- (int) open;
- (int) close;
```

The sharedManager: is a class method used to gain an instance of a singleton. The open and close methods respectively open and close the database. Each time you open the database, you should pair it with a close method. You can also provide dealloc: and automatically close the database if it is open. I will leave that for you to do as extra credit. :]

Switch to **FMDBManager.m**, and add the following properties:

```
@property (strong, nonatomic) NSString * fileName;
@property (strong, nonatomic) FMDatabase * database;
```

The database property allows the object to internally have read-write access. The filename property points to the actual filename of the database.

The first method you will implement is sharedManager: which is just a standard singleton pattern in Objective-C. Add the following code:

```
+ (id)sharedManager {
    static FMDBManager *sharedMyManager = nil;
    @synchronized(self) {
        if (sharedMyManager == nil)
        {
            sharedMyManager = [[self alloc] init];
        }
    }
    return sharedMyManager;
}
```

Now, implement the init:. Add the following:

```
- (id)init {
     self = [super init];
     if (self) {
          // ADD CODE HERE
          _fileName = [fileUrl absoluteString];
     }
     return self;
}
```

In area that reads “ADD CODE HERE”, you need to get a path to the SQLite database. The database should be called “library.sqlite” and it should exist in the library directory. Use the NSFileManager class to get the URL and convert it to a string. Conversely, you can directly get the string path as well.

For the first part of your challenge, you need to implement the open and closing methods. Read the documentation for FMDB on how to open and close database instances.

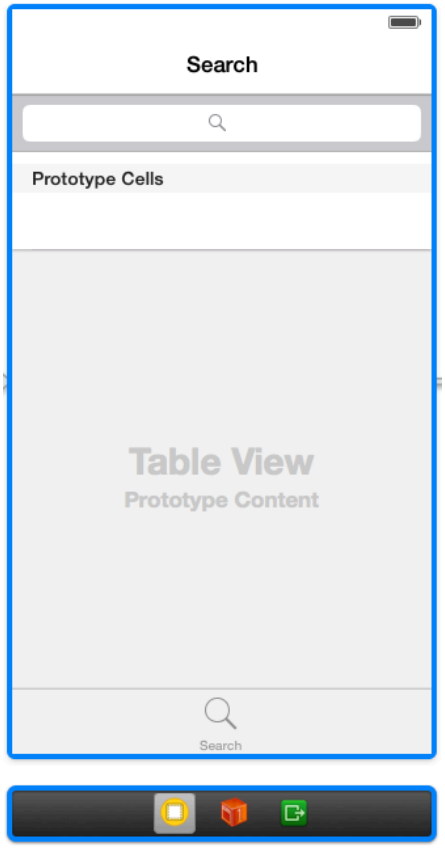
Once you have both methods working, it's time to start the main challenge.

Open **OTRLibrary.m.**

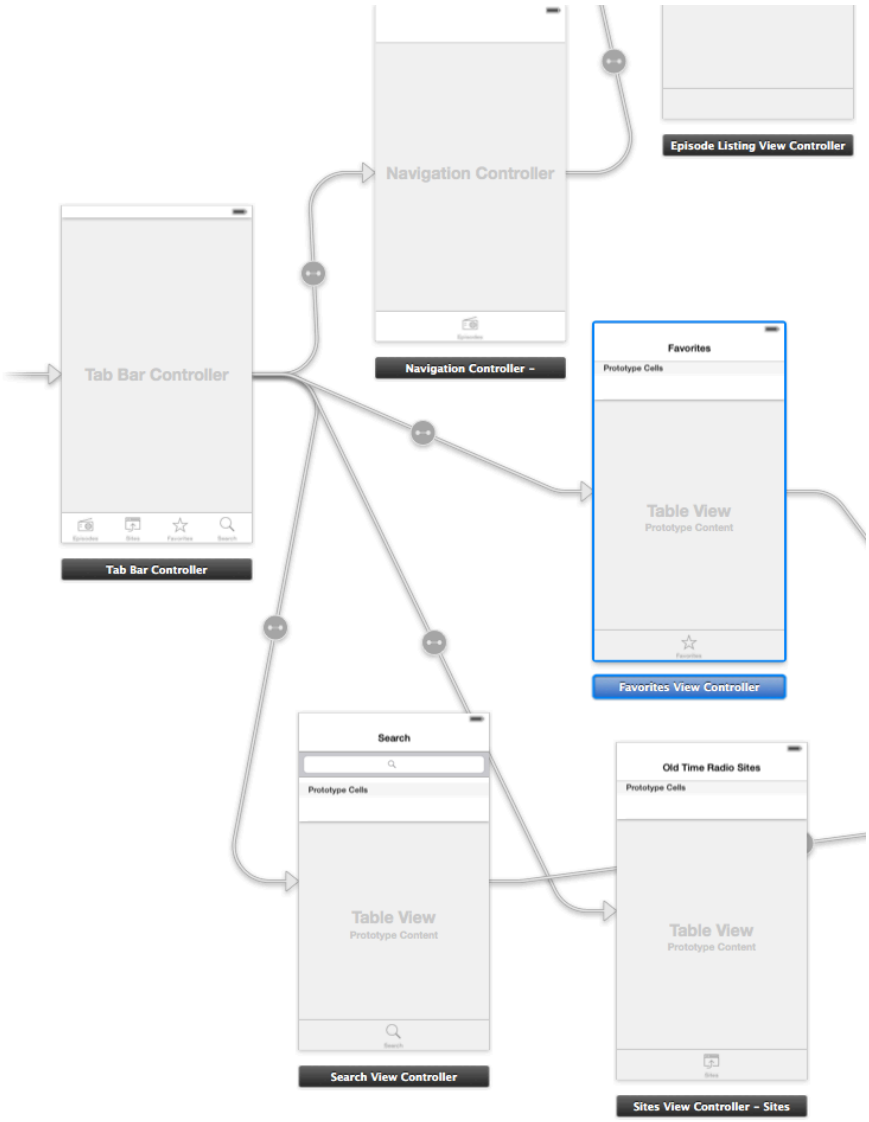
There you should see mountains of SQLite code. You challenge is to convert to FMDB. Once you have completed the conversion, the app should run without any issue.

Uber Haxx0r Challenge

While having a running old time radio is nice, but as it grows with additional shows, you will need to create a search mechanism. Your mission, if you to choose to accept it, is to first create a new view controller. This is your Search View controller and it should look like such:



This view controller should be accessible from the tab view controller so your storyboard should look something like such:



As the user types in the search box, a list of available episodes should appear. When the user taps on the row, then the PlayerViewController will open and start playing the episode.

Check out **EpisodeListingViewController.m** for a reference on how to play the old time radio episodes.