Implementing “Core Data” in iOS

Ryan Bliton
3-27-14

Abstract:
“Core Data” is a code package provided by Apple as a simple and efficient solution for memory and file management in iOS. It included pre-developed methods for ‘Save’, 'Undo' and a variety of other functions as well as a database-style storage structure.

Keywords: Memory management, disc space, save, undo, database, SQLite, Core Data, framework.
Introduction:
Some common requirements for iOS applications are editable storage and ‘save’ functionality. Apple has provided the Core Data framework in order to solve these problems in a very memory-efficient manner and as an effort saver for developers. While the framework is provided with Xcode, it must always be applied and configured by the developer. What follows is a short tutorial applying Core Data to a very simple application for storing information about books.

Part 1: Starting the project.
When using Core Data, it is best to start from a new blank project so that Xcode can apply some necessary code. It is possible to apply Core Data to an existing project but it requires a large amount of additional code and configuration by the developer.

After opening Xcode, Create a new Xcode Project, select iOS Application on the sidebar, select the Empty Application icon and click Next.

Enter a name for the app under Product Name, select the Device family (iPhone in this case) and be sure to check the Use Core Data box. The other fields are optional.

Select the folder in which to store the project and click Create.
When “Use Core Data” is checked, Xcode includes a premade data model and the Core Data framework (both selected above).

The code for the Core Data Stack is also added to the AppDelegate.m file as shown above.

The next step is to make Storyboard file by choosing New, File, User Interface under iOS, Storyboard, and then clicking Next. Choose iPhone device family, click Next again, and then name the storyboard (BookStoryboard for this case) and click Create.
Next, view AppDelegate.m and delete the contents the function “-(BOOL)application:(UIApplication *)application...” with the exception of the line “Return YES”. This will allow the storyboard to be assigned as the only interface. Afterward, click on the BookApp icon on the left, and in the General tab under Deployment Info set Main Interface to BookStoryboard as shown below.

Next click on BookStoryboard.storyboard in the sidebar and drag a TableViewController and a normal ViewController from the Object Library in the bottom right corner into the storyboard frame.
Next, select the TableViewController, click on **Editor** on the top menu bar, select **Embed In**, and then **Navigation Controller**. This is needed in order to attach a navigation bar to the TableViewController, as well as to provide a defined path for movement of data between screens.

Now zoom in on the TableViewController, double-click above “Prototype Cells” and add the title “Books”. Then drag a **Bar Button Item** object from the object library to the title bar. Select this button and in the 4th tab of the identifier window find the **Identifier** selector under **Bar Button Item** and set it to **Add**. This will allow for new entries (books) to be added to the table. Also click on the table cell below the “Prototype Cell” and set its **Identifier** to **Cell**.
After this, select the new ‘plus sign’ bar button and **Control-Click** and drag it to the other view controller. Click on the icon that appears on the connecting line and set the segue type to **modal**.

Now **embed** the other view controller in a **Navigation Controller** through the **Editor** menu. Also add the title “**Add Book**” to the view controller.
Next, add label objects and text box objects from the object library as well as a bar button to the navigation bar of the Add Book controller. Set the labels as shown below and set the button Identifier to Cancel.

![Image of Add Book interface]

Now click New File and create a Cocoa Touch Objective-C class named "BooksTableViewController", subclass of UITableViewController and separately, a Cocoa Touch Objective-C class named "AddBookViewController" subclass of UIViewController.

![Image of file creation]

![Image of file creation]
Now assign the “Books” and “AddBook” storyboard controllers to the matching class files by selecting each controller and clicking on the third tab in the window to the right and setting the **Custom Class** to the appropriate class file.

Next **Control-Click** and drag each text field in the “AddBook” controller into AddBookViewController.h, keeping **Connection** as **Outlet** and set the **Name** of the first one to “authorName” and the second to “bookTitle”. Click **Connect** to add the code. Then **Control-Click** and drag the “Cancel” button into the .h file and set **Connection** to **Action** and the **Name** to “cancel” and **Type** to UIBarButtonItem.
Now Create another New Objective-C class file named “CoreViewController” subclass of UIViewController. Then open CoreViewController.h and add the following code just below "@interface":

-(void)cancelAndDismiss;

Then, in CoreViewController.m, add the following code just above the "@end" statement:

-(void)cancelAndDismiss {

    [self dismissViewControllerAnimated:YES completion:nil];
}

Now add the following code to AddBookViewController.h just below the other "#import" statement:

#import "CoreViewController.h"

Lastly add the following code to AddBookViewController.m inside the curly brackets of – (IBAction)cancel...:

[super cancelAndDismiss];

At this point the app can be simulated and the navigation between the pages and the text fields will all function properly.