Tutorial - Part 1
Data driven app using UINavigationController and UITableViewController

Step 1 - Create the App

1. Create new project using the “Single View Application” template and click “Next”
2. Enter the following information in the options dialogue box.
    a. Product Name: **DataBasedApp**
    b. Company Identifier: **ca.echomobile**
    c. Class Prefix: **EM**
    d. Device Family: **iPhone**
    e. Use Storyboard: **unchecked**
    f. Use Automatic Reference Counting: **checked**
    g. Include Unit Tests: **unchecked**

Step 2 - Add a UINavigationController

3. This will replace the default *rootViewController* with a UINavigationController.
4. In the ***didFinishLaunchingWithOptions*** selector in ***EMAppDelegate.m***, after the line where *EMViewController* is created, add the following code to create the UINavigationController.

```
UINavigationController *tmpNavController = [[UINavigationController alloc]
initWithRootViewController:self.viewController];
```

5. On the following line, replace the *rootViewController* with the newly created UINavigationController

```
self.window.rootViewController = tmpNavController;
```

RUN: If you run the app at this point, you should see a navigation bar at the top of the app.

Step 3 - Setup the first view.

6. In the *viewDidLoad* selector of *EMViewController.m* add the following code.

```
self.title = @"Menu";
```

RUN: If you run the app at this point, the new title will appear in the navigationController.

Step 4 - Add a Second view.

7. Add a new file to the project. (**iOS->Cocoa Touch->UIViewController**).

8. Call the file “EMSecondViewController”, and **make sure it is a subclass of UIViewController**.
9. Select *EMSecondViewController* and drag a button onto the view.
10. Before we go any further, let’s give our button some graphics. From the folder with the completed project, grab the “button_default.png” and “button_pressed.png” files and add them to a folder called Resources in your project. Add this entire folder to your XCode project by selecting *File->Add Projects to...*
11. Open *EMSecondViewController.xib,* select the button, and open the *Attributes Inspector*.
12. Set the button “type” to *Custom*.
13. With the *State Config* set to “Default”, set the *Title* to **Show store list**, and set the *Background* to **button_default.png**, and set the *Text Color* to white.
14. With the *State Config* set to “Highlighted”, set the *Background* to **button_highlighted.png**. (Note: there is no need to set the *Title* for the Highlighted state, as the button will use the default title if one isn’t set for the Highlighted state.

15. Repeat steps 9-14 and add a button to *EMViewController.xib* with the title of “Show second view”.

NOTE: These graphics are open source and available from premiumpixels.com

Step 5 - Connect the two views.

NOTE: At this point if you were to run the app, you’d notice that when you click on the button in the first view, nothing happens!

16. Open *EMViewController.xib* and select the button.
17. Open the *Assistant Editor.*
18. Select the button, open the *Connections Inspector,* and drag the Pick Whip for **Touch Up Inside** over to the code are for *EMViewController.h.* Call the action “showSecondView”.
19. Close the *Assistant Editor*.
20. Open *EMViewController.h* and import the class for the second view controller by adding the following line at the top of the file, just under the other ***import*** statement.

    ```
    #import "EMSecondViewController.h"
    ```

21. Open *EMViewController.m* and look for the newly created *showSecondView* selector. Add the following code to that selector.

    ```
    EMSecondViewController *tmpTableViewController = [[EMSecondViewController alloc] init];
    [self.navigationController pushViewController:tmpTableViewController animated:TRUE];
    ```

RUN: If you run the app at this point, you should be able to click on the button and see the second view appear.

BONUS: Step 6 - A title for the second view controller

22. Repeat Step 6 to add a title to the second view controller.

Step 7 - Add a table view controller.

23. Create a new file for this project. (iOS->Cocoa Touch->UIViewController subclass).
24. Call the file "EMTableViewController", and **make sure it is a subclass of UITableViewController**.
25. Remember that button you created in EMSecondViewController.xib? Repeat steps 16-21 to connect that button to an action called "showTableViewController".
26. For Step 20, you'll be importing EMTableViewController.h.
27. For Step 21, add the following code instead.

```
TableViewController *tmpTableViewController = [[TableViewController alloc] init];
[self.navigationController pushViewController:tmpTableViewController animated:TRUE];
```

RUN: If you run the app at this point, you should be able to click through the first two buttons to see a table appear. But there is no content!

Step 8 - Add content to the table view controller.

28. In *EMTableViewController.h,* add the following code just above the **@end** directive.

```
@property (nonatomic, strong) NSArray *tableContent;
```

29. In *EMTableViewController.m*, add the following code just below the **@implementation** directive.

```
@synthesize tableContent;
```

30. Now let's create the array. In EMTableViewController, in the *initWithStyle* selector, just below the line where it says ***//*Custom initialization**, add the following code.

```
self.tableContent = [NSArray arrayWithObjects:
                     @"Toronto",
                     @"Ottawa",
                     @"Montreal",
                     @"Vancouver",
                     @"Winnipeg",
                     @"Calgary",
                     @"Halifax",
                     @"St. John",
                     @"Whitehorse",
```

```
                @"Yellowknife",
                @"Iqaluit",
                nil];
```

31. Modify the *numberOfSectionsInTableView* selector to return 1.
32. Modify the numberOfRowsInSection selector to return

```
        [self.tableContent count];
```

33. In the *cellForRowAtIndexPath selector,* Immediately after the line that says “Configure the cell”, add the following line of code.

    ```
    cell.textLabel.text = [self.tableContent objectAtIndex:indexPath.row];
    ```

34. You’re done!