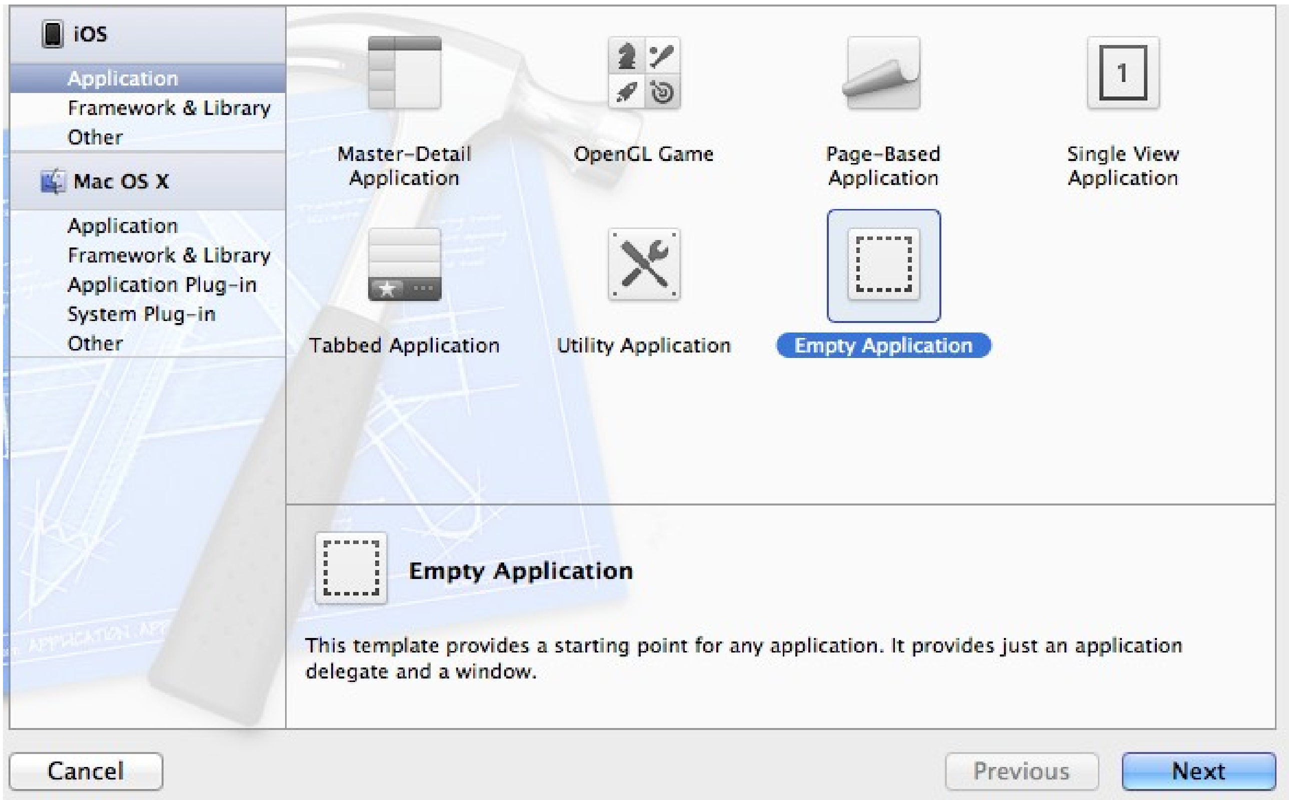
How to swipe between views in iOS 5

The source code can be found at Github, along with screenshots in a pdf :

1. Set up a healthy project

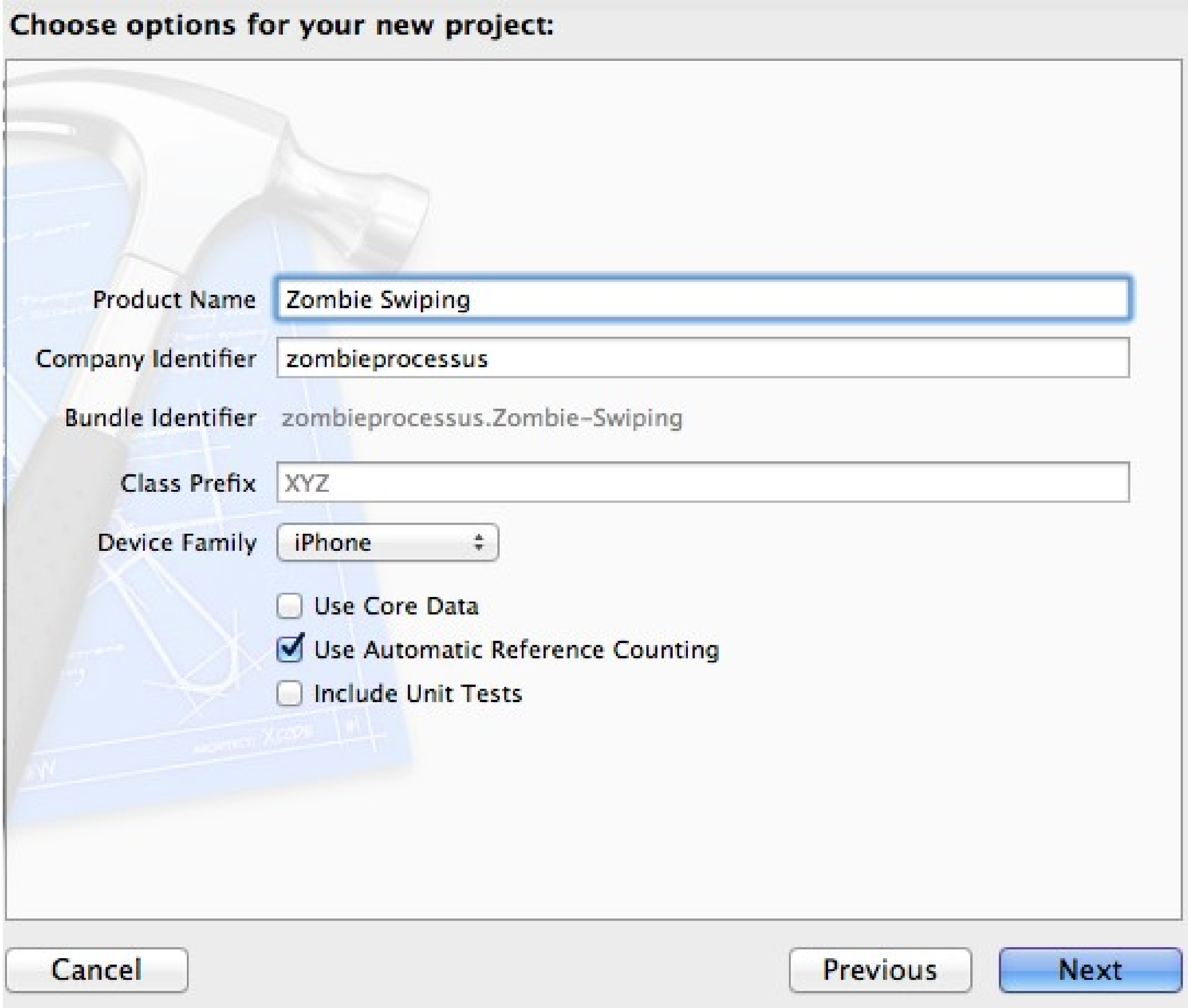
In File > New > New Project, choose "Empty Application". Press Next.



Empty Application

This template provides a starting point for any application. It provides just an application delegate and a window.

Choose whatever product name you want. For example: Zombie Swiping.
We won't need Core Data for this tutorial so you can uncheck it. Leave "Use Automatic Reference Counting" checked. Press Next.



Choose options for your new project:

Product Name: Zombie Swiping

Company Identifier: zombieprocessus

Bundle Identifier: zombieprocessus.Zombie-Swiping

Class Prefix: XYZ

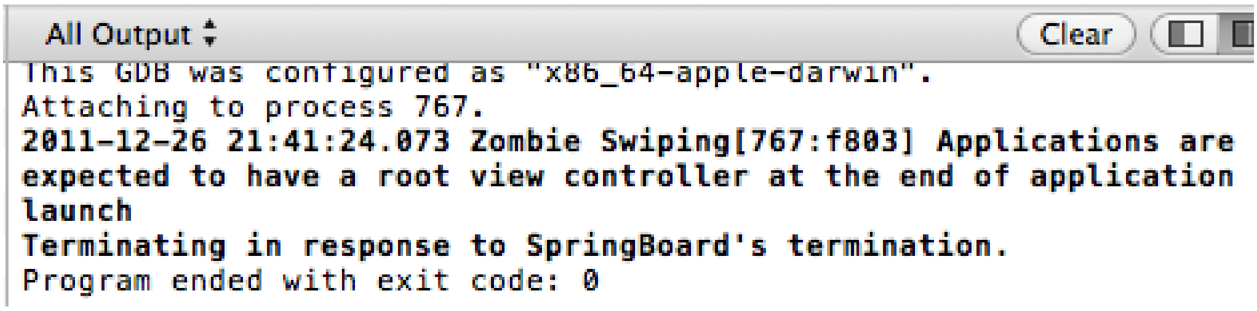
Device Family: iPhone

☐ Use Core Data

☑ Use Automatic Reference Counting

☐ Include Unit Tests

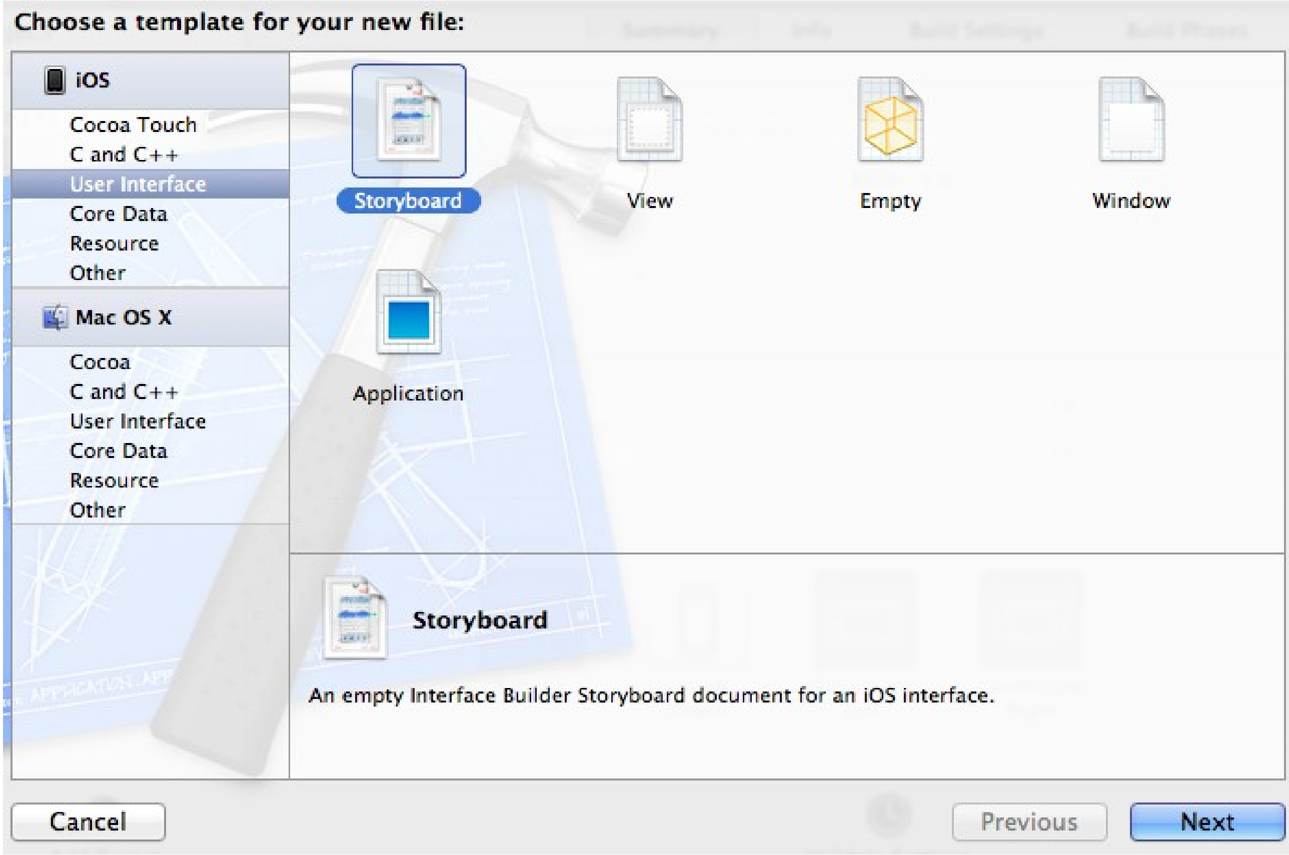
Create the project wherever you want. Your iOS 5 project is set! If you Build and Run the project at this point you should see a white window and the analyser will moan about the necessity of setting a Root View Controller. It's exactly what we are going to do now!



```
This GDB was configured as "x86_64-apple-darwin".
Attaching to process 767.
2011-12-26 21:41:24.073 Zombie Swiping[767:f803] Applications are
expected to have a root view controller at the end of application
launch
Terminating in response to SpringBoard's termination.
Program ended with exit code: 0
```

2. Add a storyboard

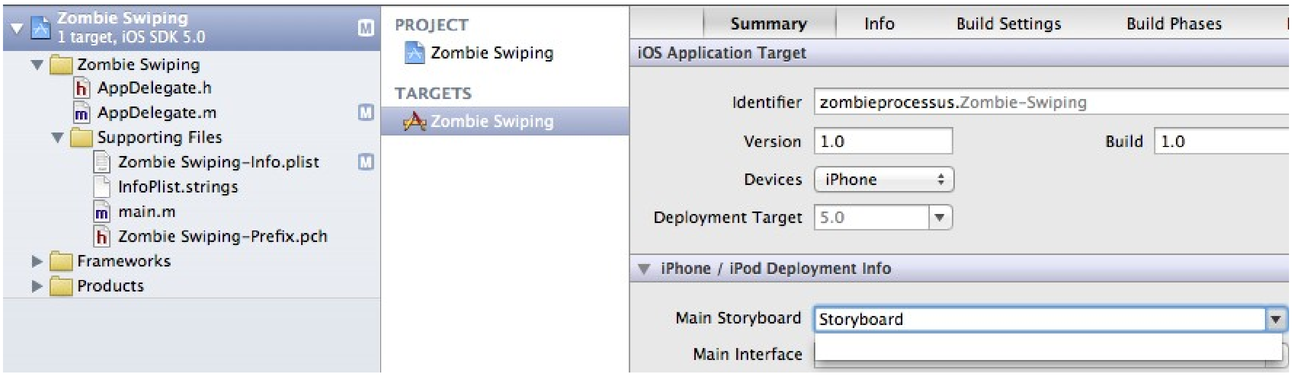
We're going to start by adding a storyboard: the new iOS way for creating views and linking between them. To do that just right click on "Zombie Swiping" in the hierarchy tree, select "New File..." and then select "Storyboard" in the user interface part. Press next.


Choose a template for your new file:

An empty Interface Builder Storyboard document for an iOS interface.

The default Storyboard name will do. Create the file and you'll see the empty story board appear on your screen. Hooray!

We now need to define it as the main storyboard so that the program knows where to look for the views. Click on the top file in the hierarchy tree called "Zombie Swiping". In the new view that appears in the right window click on Zombie Swiping under "Targets". In the "Main Storyboard" section select your newly create storyboard. Now the program knows which file to look for your views. Let's create them already!



Identifier zombieprocessus.Zombie-Swiping

Version 1.0

Build 1.0

Devices iPhone

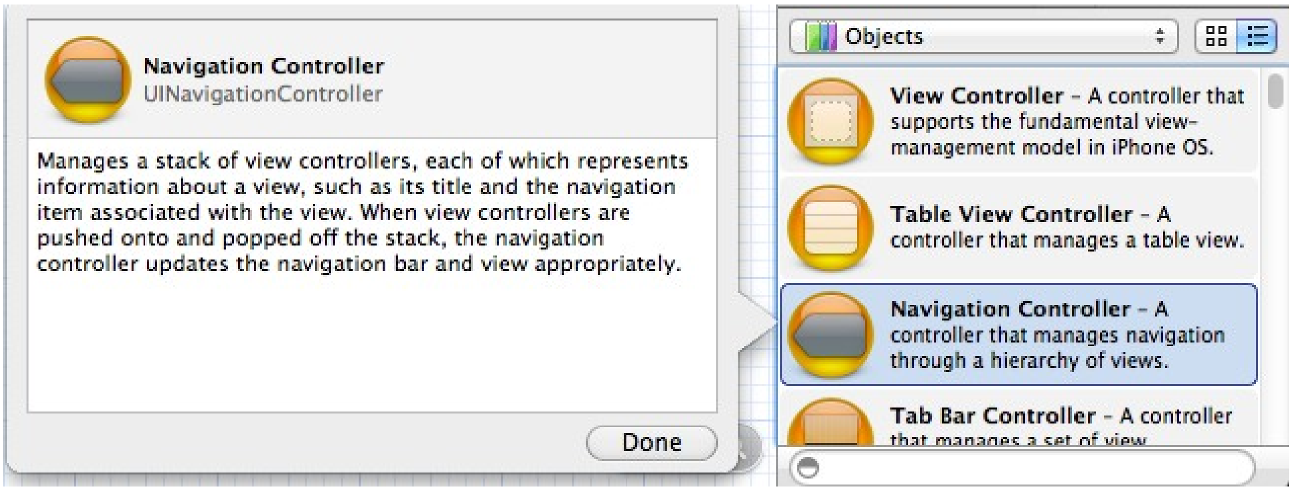
Deployment Target 5.0

Main Storyboard Storyboard

Main Interface

3. Create 2 views in your storyboard

Click on your storyboard. You should see a basic background and nothing else. let's fill this emptiness with a nice navigation controller. Select the navigation controller in the right part of the screen and drag-and-drop it in your storyboard. Tadaa! You now have a navigation controller linked to a view controller.



Navigation Controller
UINavigationController

Manages a stack of view controllers, each of which represents information about a view, such as its title and the navigation item associated with the view. When view controllers are pushed onto and popped off the stack, the navigation controller updates the navigation bar and view appropriately.

View Controller - A controller that supports the fundamental view-management model in iPhone OS.

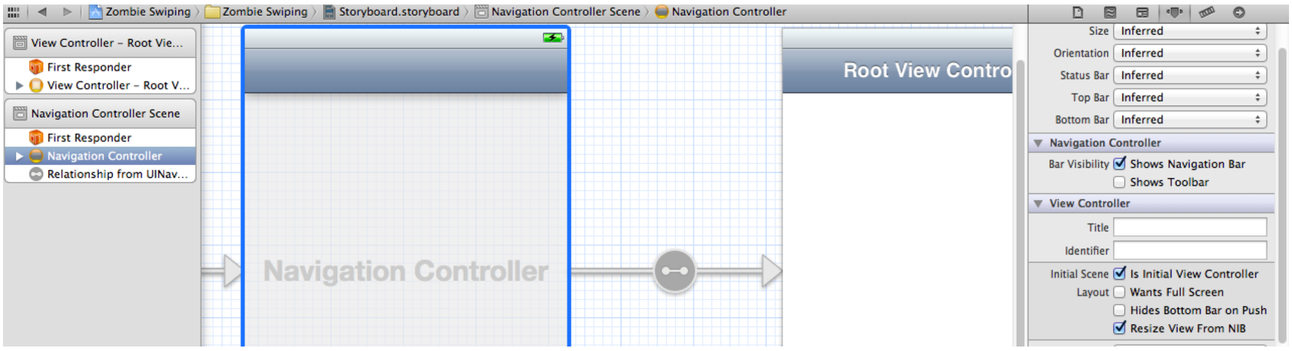
Table View Controller - A controller that manages a table view.

Navigation Controller - A controller that manages navigation through a hierarchy of views.

Tab Bar Controller - A controller that manages a set of view

But that's not enough. We need to tell iOS that the navigation controller represents our point of entry in the storyboard.

Click on the navigation controller. In the right part where its attributes are displayed, click on the 4th icon to open the attributes inspector. Under the "view controller" section check the option "is Initial View Controller". You should now see an arrow appear to the left of your Navigation Controller, confirming indeed that this is now your entry point in the storyboard, and the first view displayed.



Zombie Swiping > Zombie Swiping > Storyboard.storyboard > Navigation Controller Scene > Navigation Controller

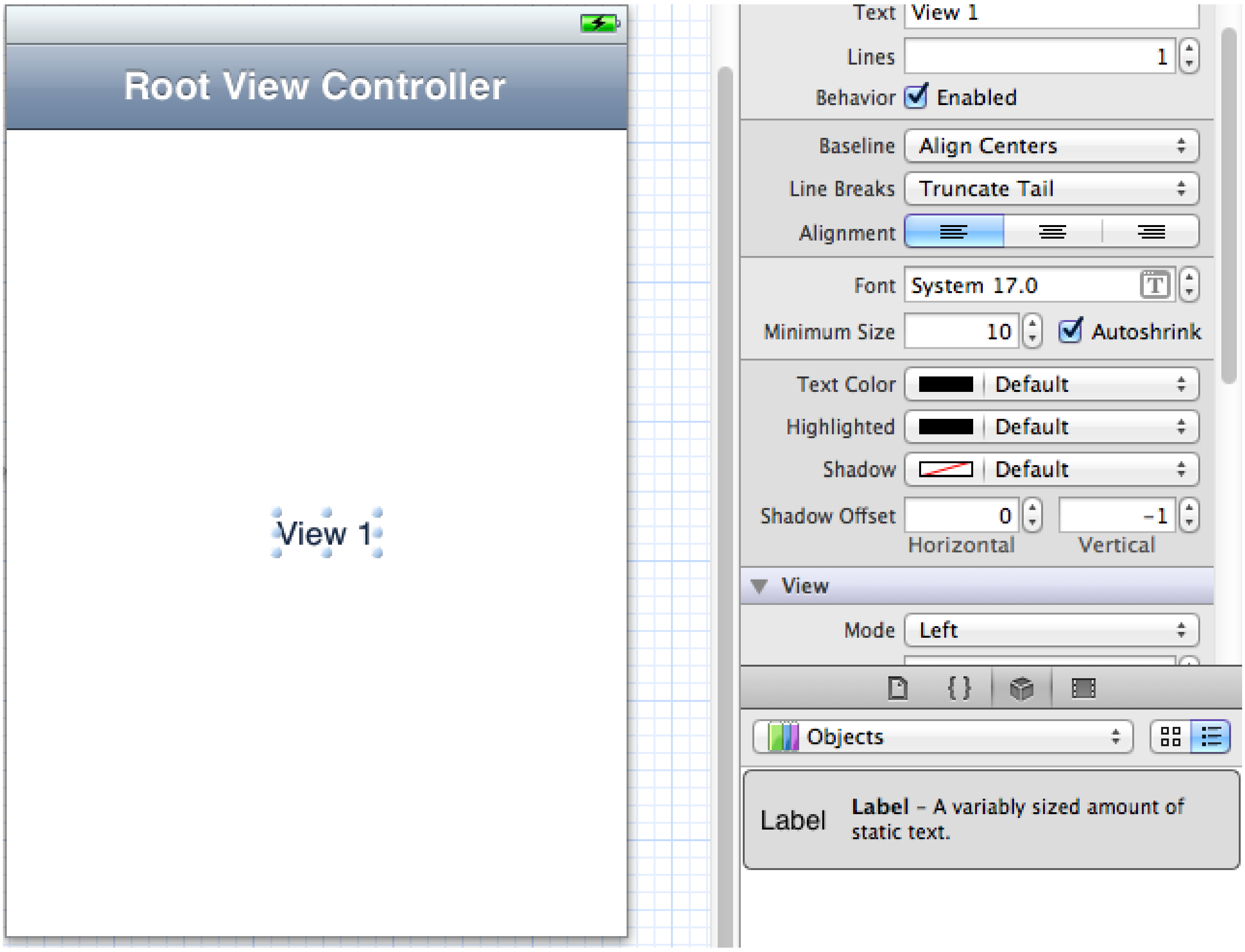
- View Controller – Root Vie...
  - First Responder
  - View Controller – Root V...
- Navigation Controller Scene
  - First Responder
  - Navigation Controller
  - Relationship from UINav...

Size: Inferred
Orientation: Inferred
Status Bar: Inferred
Top Bar: Inferred
Bottom Bar: Inferred

Navigation Controller
Bar Visibility ☑ Shows Navigation Bar
□ Shows Toolbar

View Controller
Title
Identifier
Initial Scene ☑ Is Initial View Controller
Layout □ Wants Full Screen
□ Hides Bottom Bar on Push
☑ Resize View From NIB

In order to check that the view is indeed loaded when building the project, add a label in the first view. Let's call it "View 1".



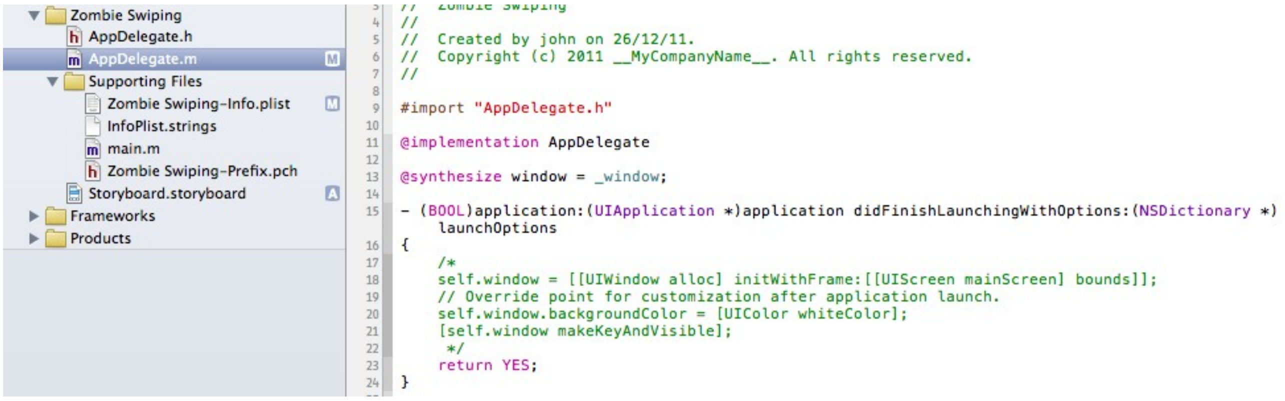
Text: View 1
Lines: 1
Behavior: ☑ Enabled
Baseline: Align Centers
Line Breaks: Truncate Tail
Alignment
Font: System 17.0
Minimum Size: 10 ☑ Autoshrink
Text Color: Default
Highlighted: Default
Shadow: Default
Shadow Offset: 0 Horizontal, -1 Vertical

View
Mode: Left

Label **Label** – A variably sized amount of static text.

Now Build and Run the project, and you should see… a white window and no label! Oh no, what happened? Actually, do you remember the first white window we had upon the first launch? It's exactly the same one being relaunched because of some piece of code in AppDelegate.m Indeed if there wasn't this code we wouldn't have had a white window upon launching the project but a black screen!

So let's select AppDelegate.m and comment/delete the first few lines until "Return YES;" Build and Run the project again. Your first view should now be displayed. Well done!



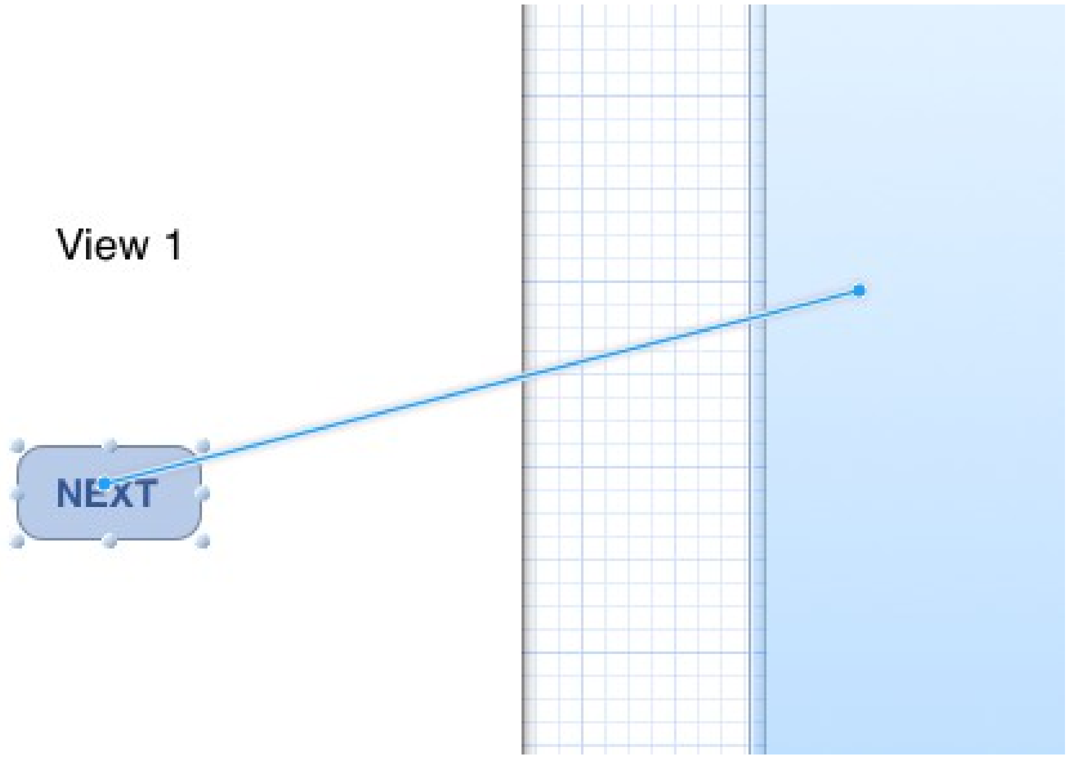
```
//  Zombie Swiping
//
//  Created by john on 26/12/11.
//  Copyright (c) 2011 __MyCompanyName__. All rights reserved.
//

#import "AppDelegate.h"

@implementation AppDelegate

@synthesize window = _window;

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)
    launchOptions
{
    /*
    self.window = [[UIWindow alloc] initWithFrame:[[UIScreen mainScreen] bounds]];
    // Override point for customization after application launch.
    self.window.backgroundColor = [UIColor whiteColor];
    [self.window makeKeyAndVisible];
     */
    return YES;
}
```

Now go back to the story board and drag-and-drop a View Controller to the right of our first view. Then add a label in it. Let's call it "View 2".

3.5 Link the views with a button

Now in order to link these 2 views, let's add a button in View 1. Drag-and-drop it from the usual list under the View

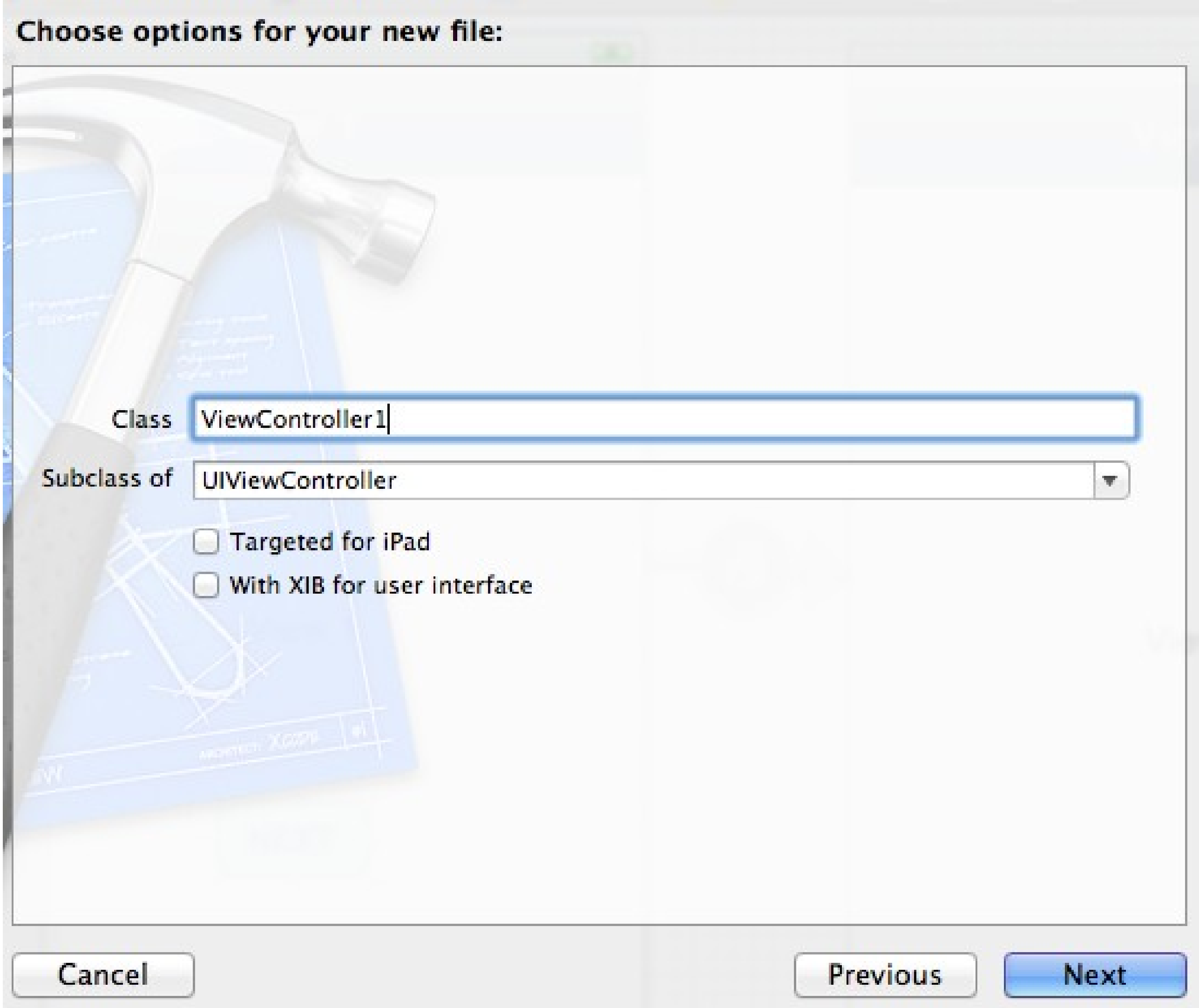
1 label. Now, Ctrl+click on the button, hold your click and drag the now-appearing line to the next view. Release your click: In the little black window that appears select "push".



Build and Run your project, click the button. Yes! You can already navigate from view 1 to view 2 thanks to the new Segue that you have created (can you see it appearing in the list of UI elements to the left of the storyboard?). Time to make it even cooler by adding the new swipe gesture recogniser from Apple.

4. Add the code to handle segues

You can delete the button or leave it, it doesn't matter.
Now, go back to your hierarchy tree, right click on the Zombie Swiping file, select "New File" and select "UIViewControllerSubclass" in "Cocoa Touch". Call it ViewController1 and create it. Repeat this step and create another UIViewControllerSubclass called ViewController2.



Choose options for your new file:

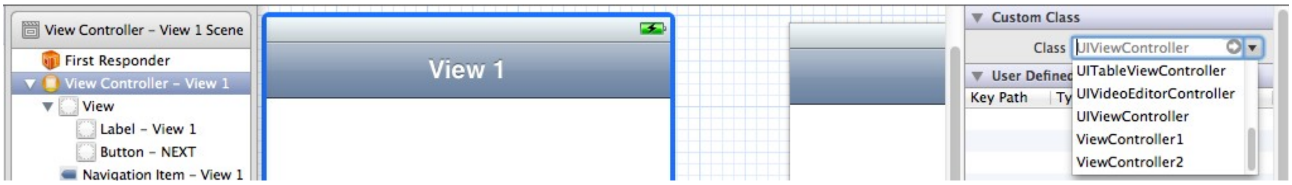
Class: ViewController1

Subclass of: UIViewController

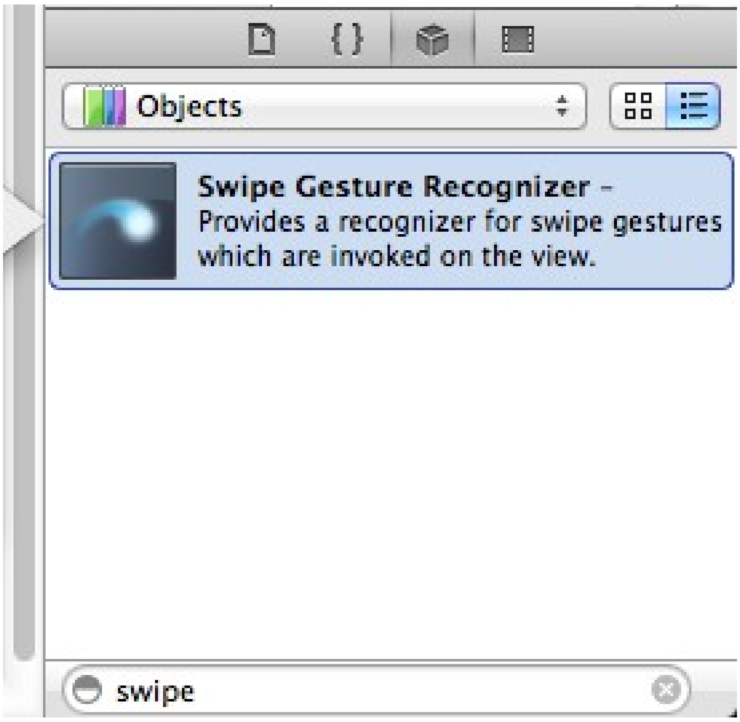
☐ Targeted for iPad

☐ With XIB for user interface

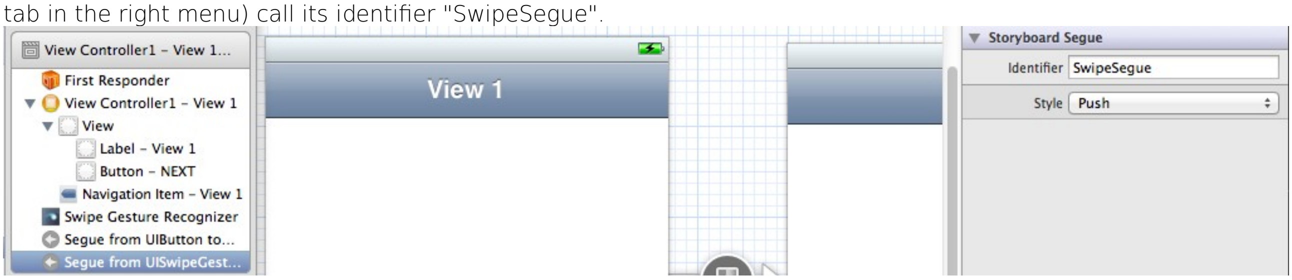
Good, now we have the files needed to handle what's happening in both our views. Let's tell these views which files control them! Go back to your storyboard, select the first view controller and in the right part select the 3rd tab (the identity inspector) and in the class list select "ViewController1". Now the first view controller knows where to look for code-handling of the different events.
Repeat this step with the second view controller (select ViewController2).



We are also going to add the swipe gesture from right to left to the first view. Select it in the list of objects and drag-and-drop it in your first view.


Objects
Swipe Gesture Recognizer - Provides a recognizer for swipe gestures which are invoked on the view.
swipe

Then Ctrl+click on it, drag the line to the next view, release the click and select "push". Now a new Segue called "Segue from UISwipeGestureRecognizer" appears in the left menu. Select it and in the Attributes inspector (4th tab in the right menu) call its identifier "SwipeSegue".



Identifier SwipeSegue
Style Push

Nicely done! Now go to ViewController1.m and in the viewDidLoad section add:

```
UISwipeGestureRecognizer *swipeLeft = [[UISwipeGestureRecognizer alloc] initWithTarget:self
action:@selector(handleSwipeLeft:)];
swipeLeft.direction = UISwipeGestureRecognizerDirectionLeft;
[self.view addGestureRecognizer:swipeLeft];
```

And further down in another section add:

```
- (void)handleSwipeLeft:(UISwipeGestureRecognizer *)recognizer {

    [self performSegueWithIdentifier: @"SwipeSegue" sender: self];
}
```

That's it! Build and Run your project, and try to swipe from right to left on the first screen. Your first view should be replaced by your second view, just like with the button.