https://github.com/GroundSpeed/SwiftlyBreakingGround

SWIFTLY BREAKING GROUND WITH IOS

Don Miller
Founder of GroundSpeed™
Co-founder of Seed Coworking
@donmiller

Don Miller | GroundSpeed™
rapid web + mobile software

SCHEDULE

Introductions

Discuss Terminology, Xcode Overview

Create Your First App: Hello World!

MVC Overview

Color Me Demo

Swift Basics and the Playground

Storyboards

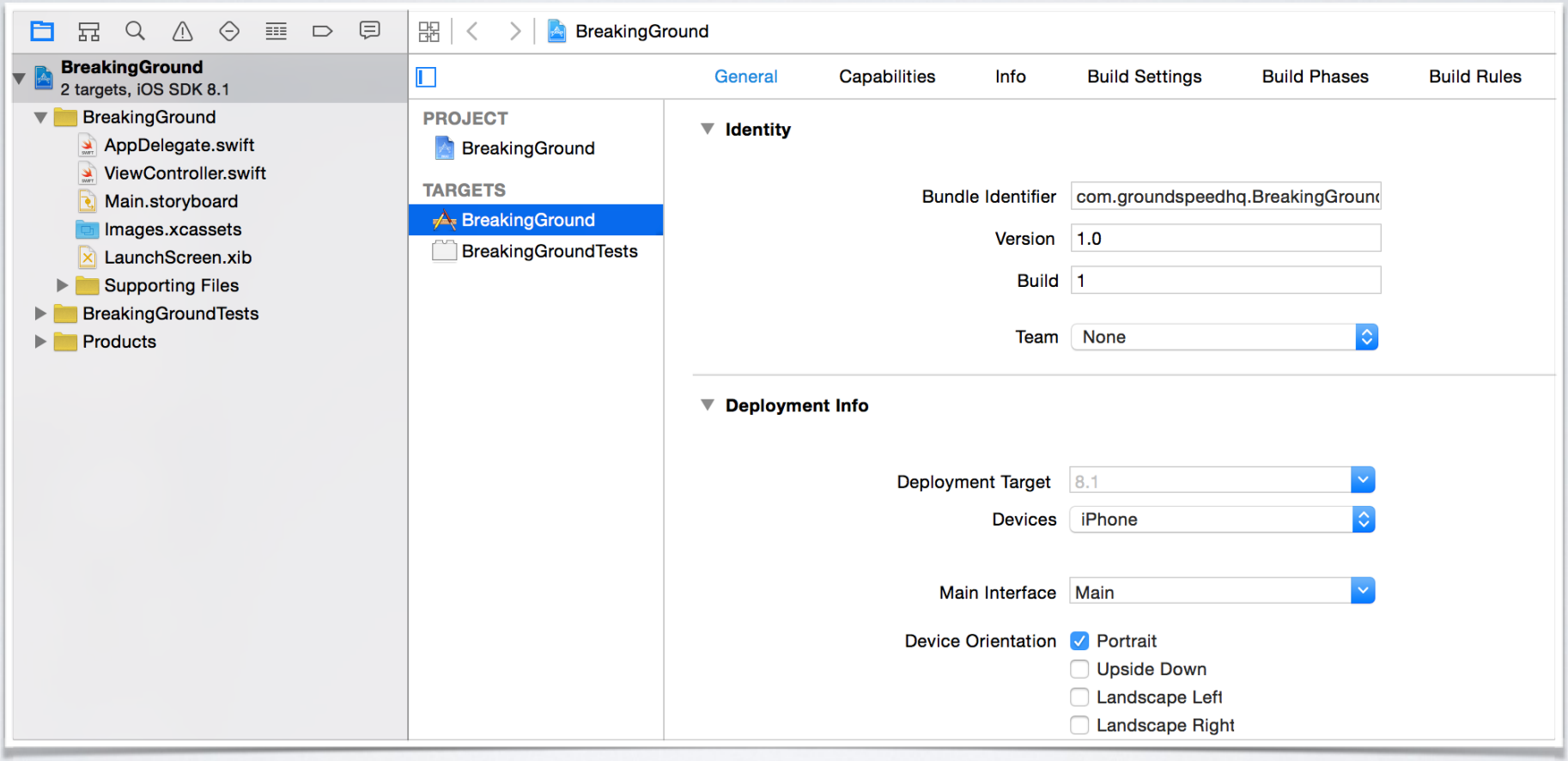
UITableViews (if we have time)

GroundSpeed™
rapid web + mobile software

TERMINOLOGY

- iOS
- iPhone
- iPad
- iPod
- MacBook
- iMac
- SSD
- SDK
- IDE
- Xcode
- OSX
- Mavericks
- Mountain Lion
- Lion
- Snow Leopard, etc.
- Retina

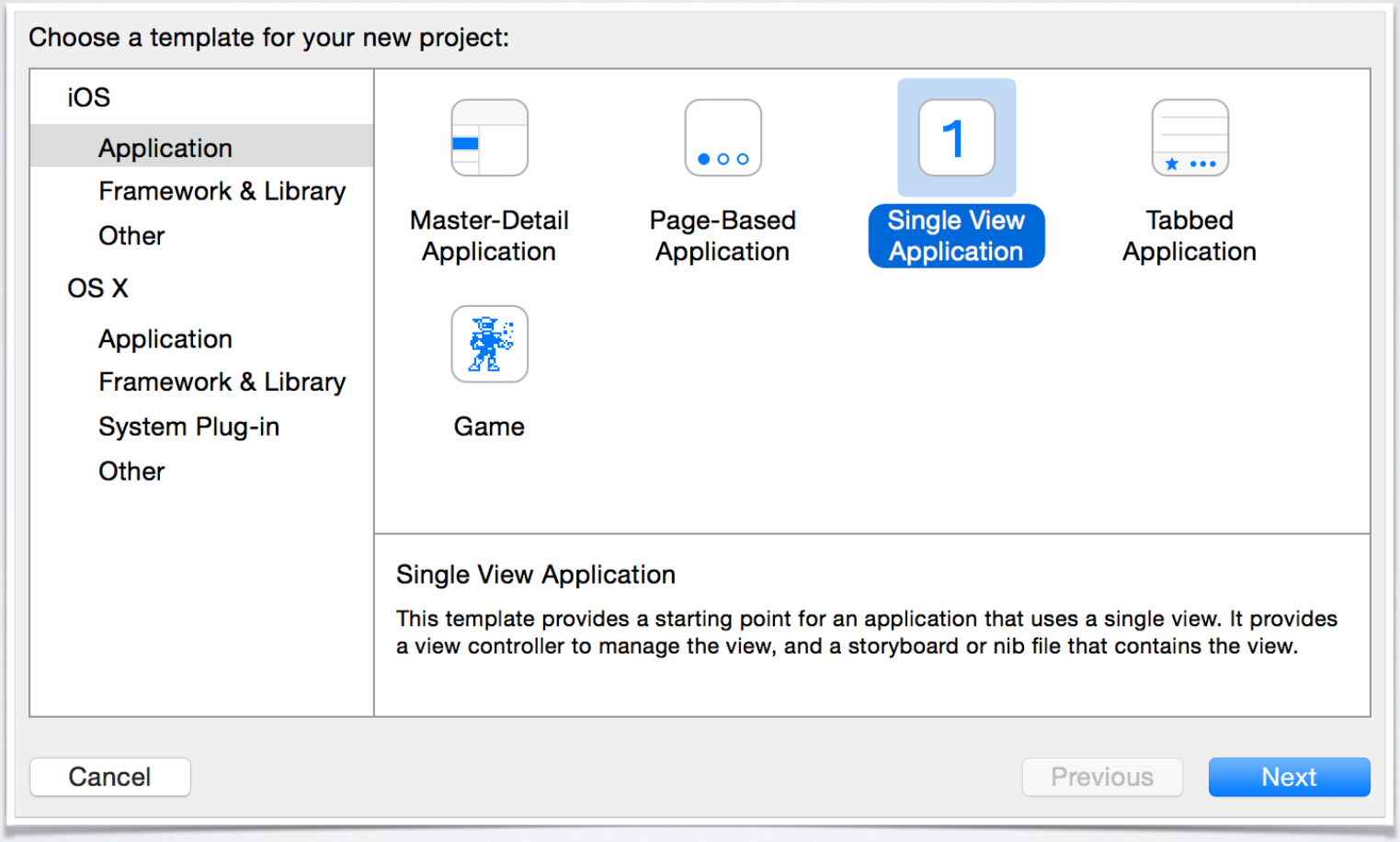
XCODE 6.1 & IOS 8.1.2



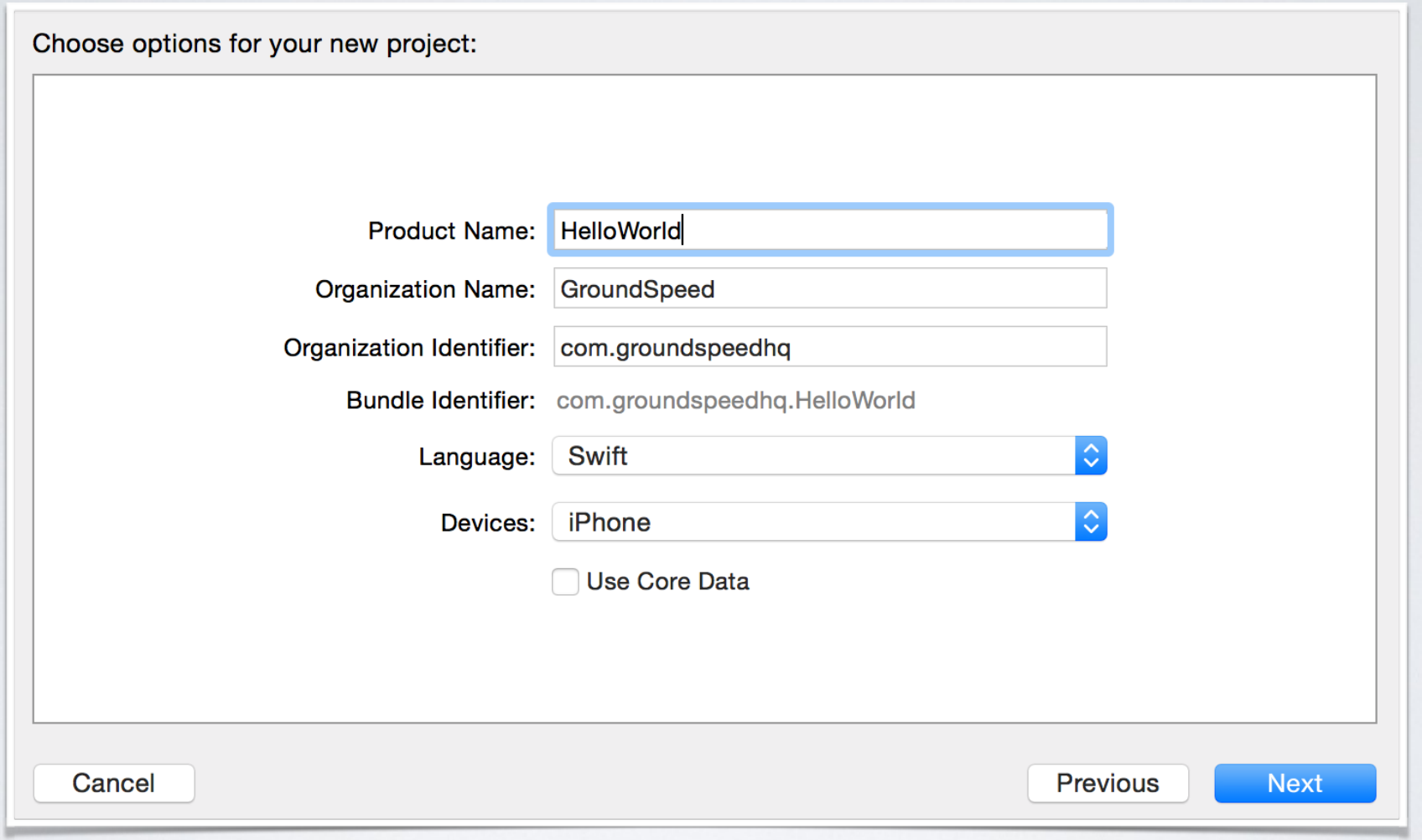
GroundSpeed™
rapid web + mobile software

HELLO WORLD!

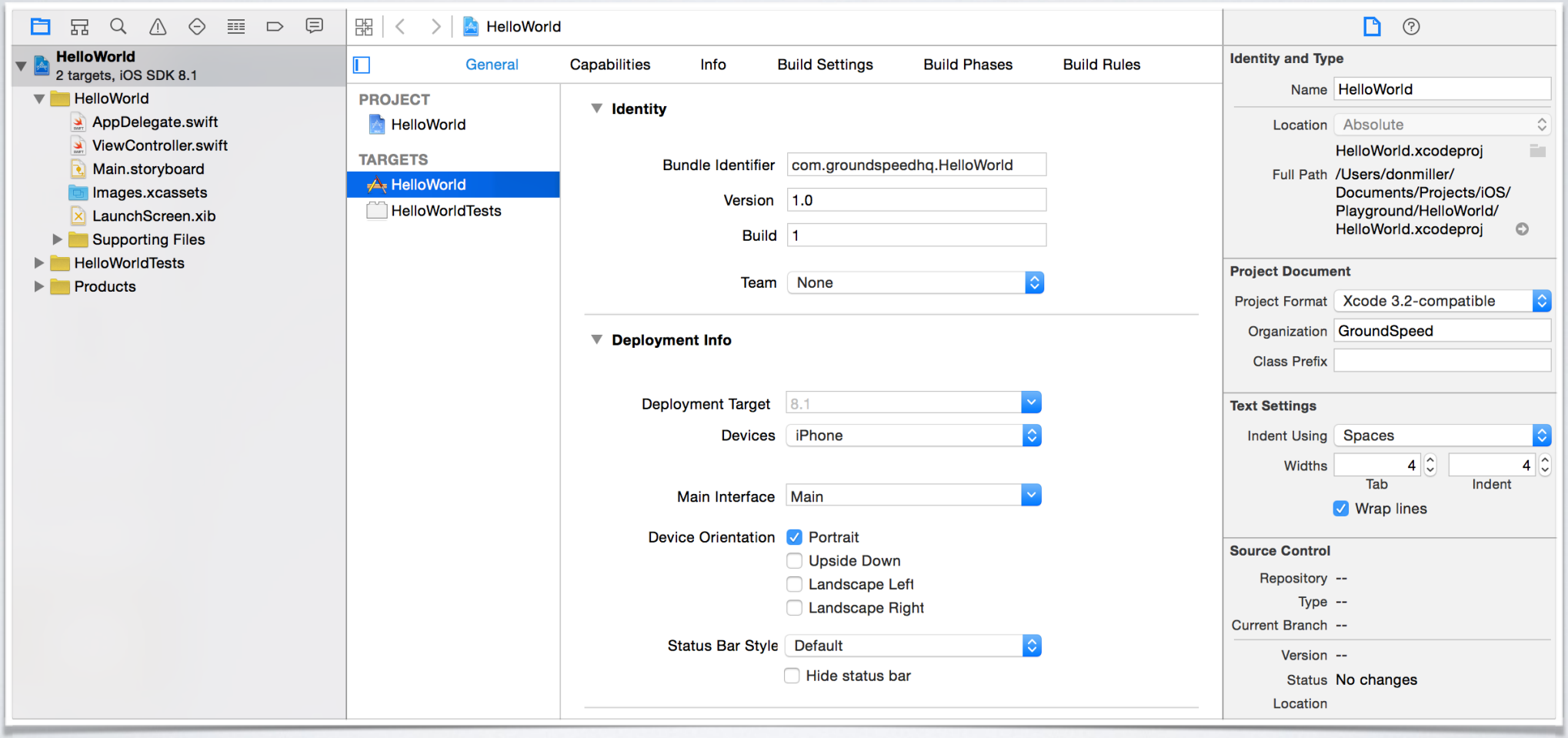
CREATE A NEW SINGLE VIEW APPLICATION



ENTER THE PRODUCT NAME

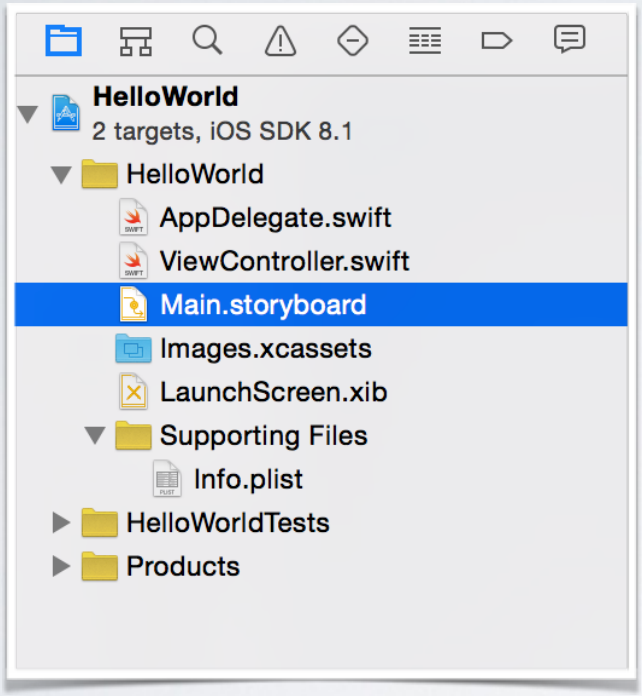


UPDATE THE GENERAL PROJECT ATTRIBUTES



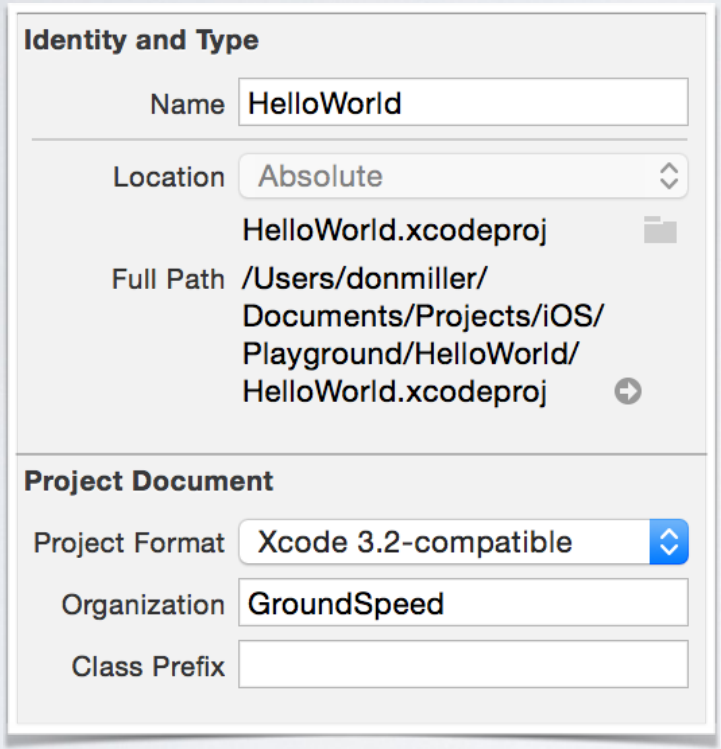
GroundSpeed™
rapid web + mobile software

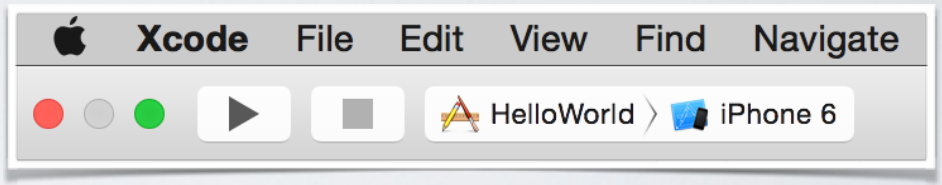
PROJECT STRUCTURE

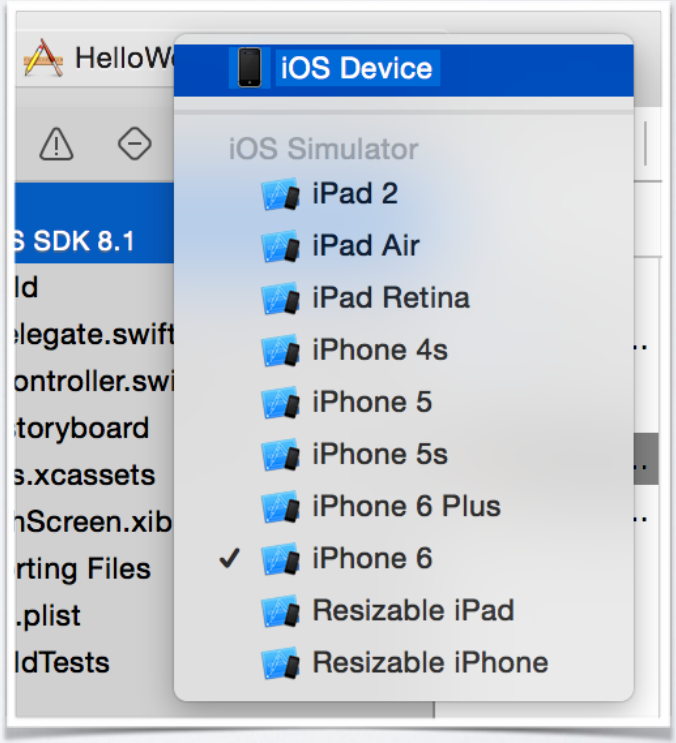


- Folders do not match physical layout on file system.
- Actually you will not find any sub folder named like Supporting Files in your physical project folder.
- The folder hierarchy in the XCode's navigator is simply a logical grouping of the resources.
- At this stage, we would be interested only on the Main.storyboard file.

FILE INSPECTOR & SCHEMA SETTINGS

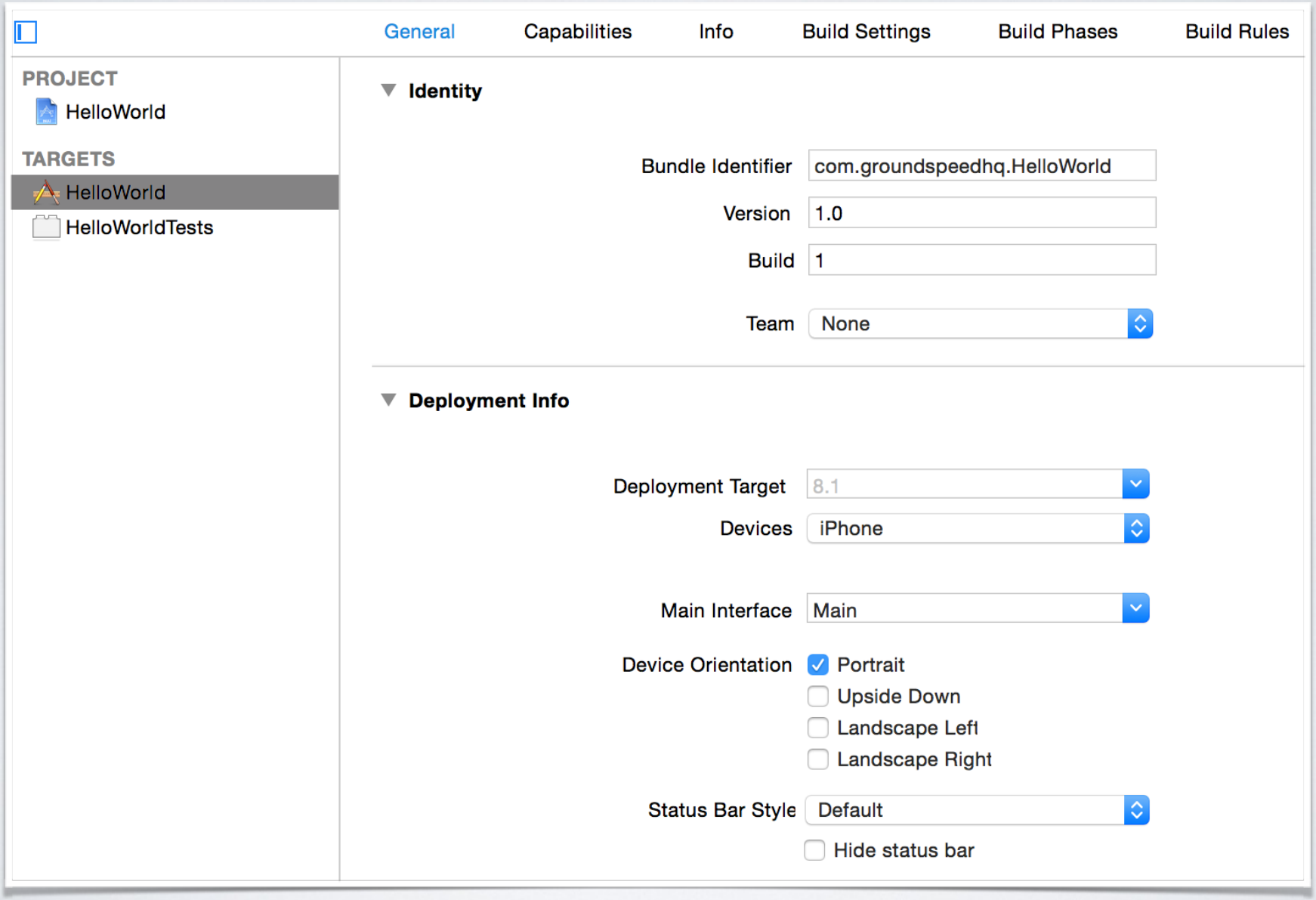




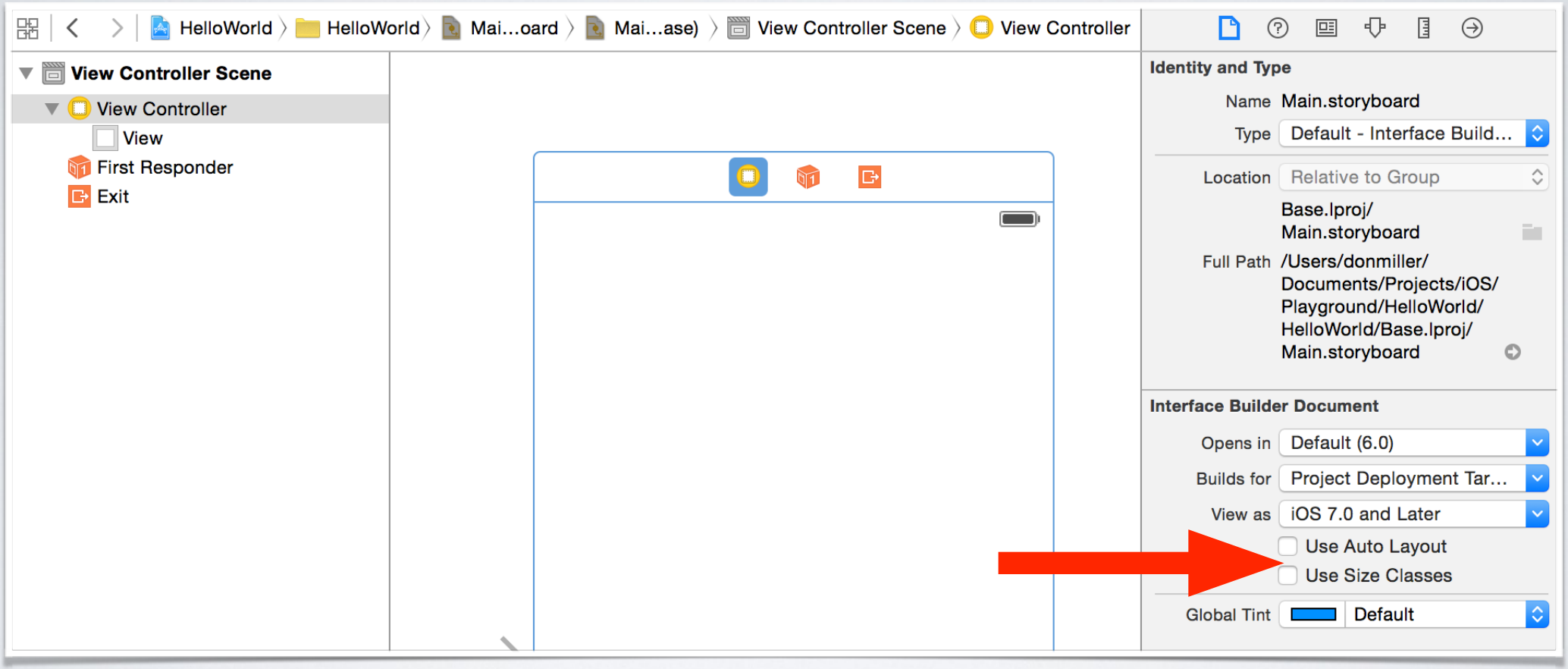


GroundSpeed™
rapid web + mobile software

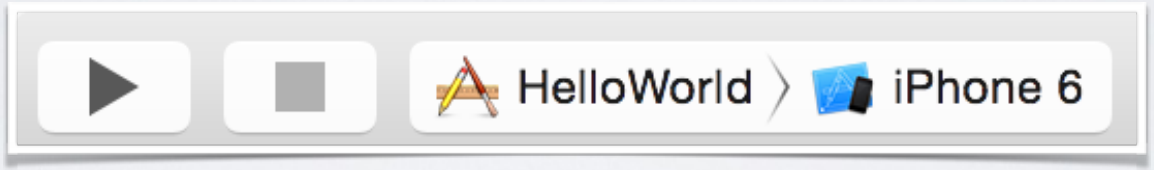
DEVICES & TARGETS



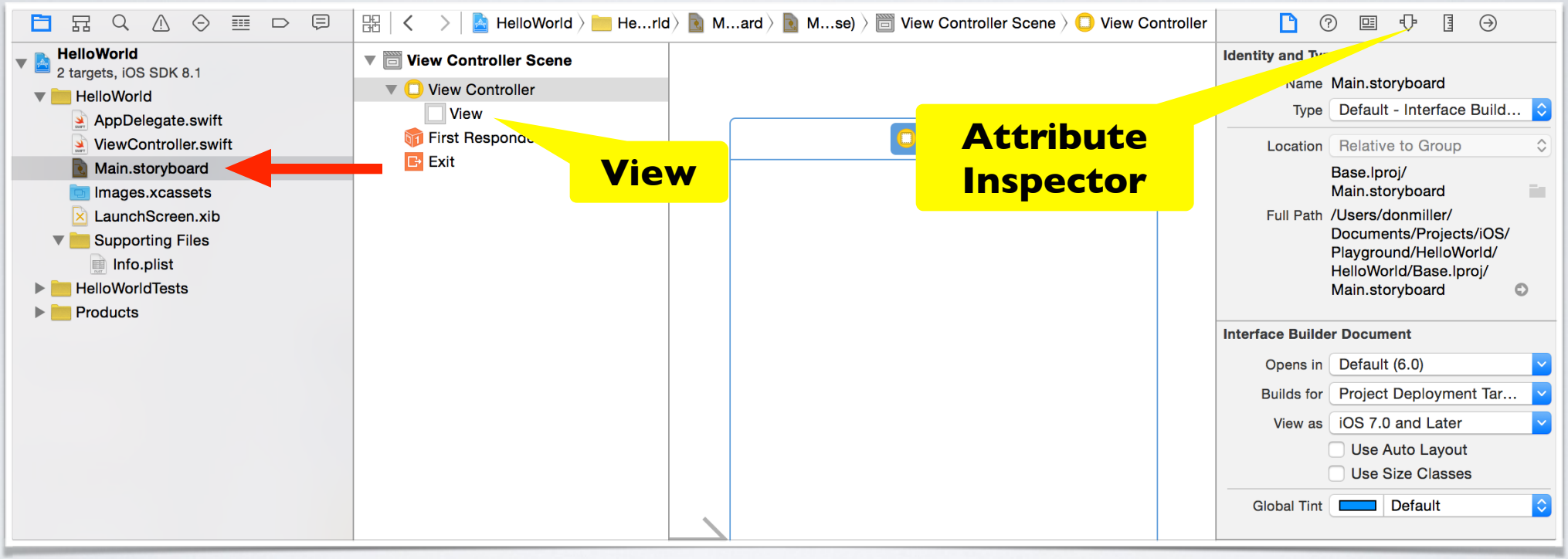
FILE OWNER, VIEW, & FIRST RESPONDER



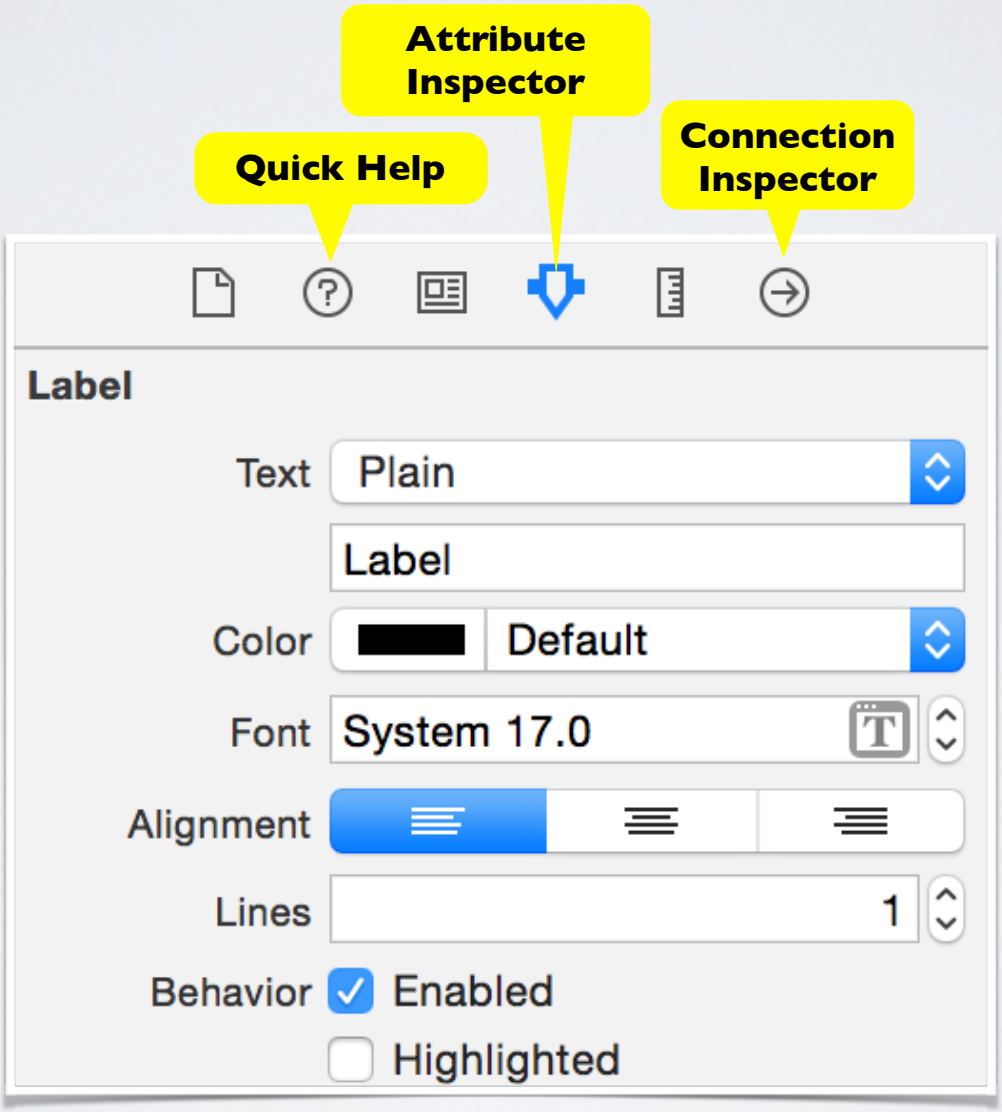
RUN THE APPLICATION BY PRESSING PLAY



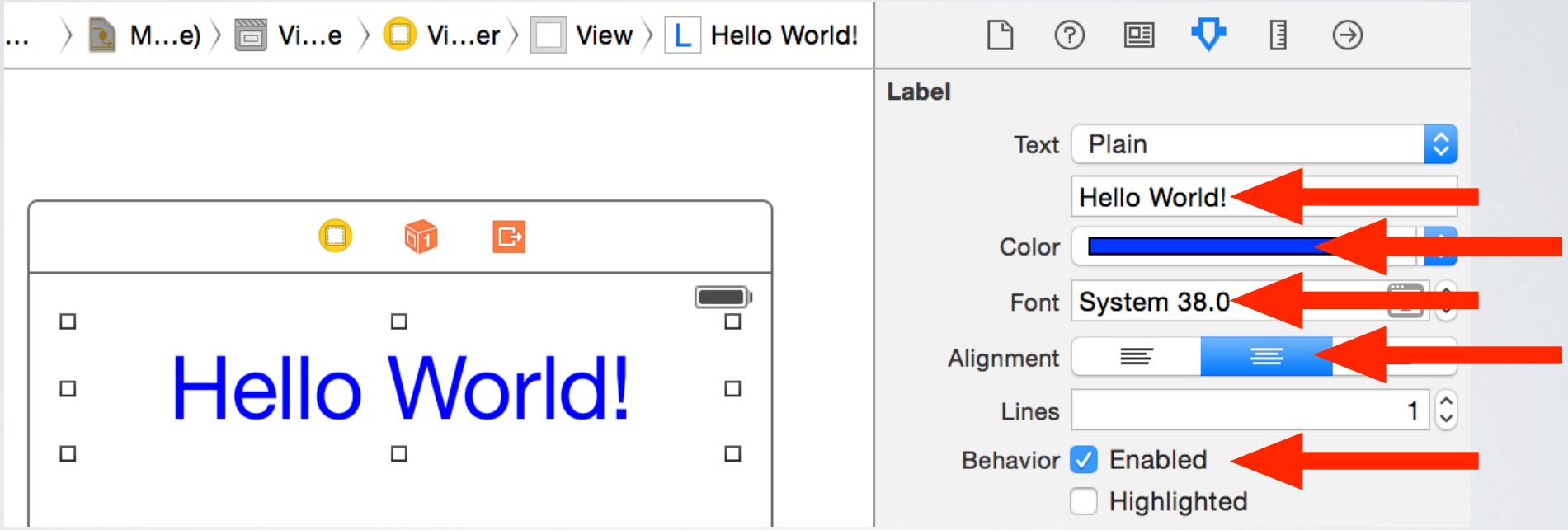
ATTRIBUTE INSPECTOR



INSPECTORS



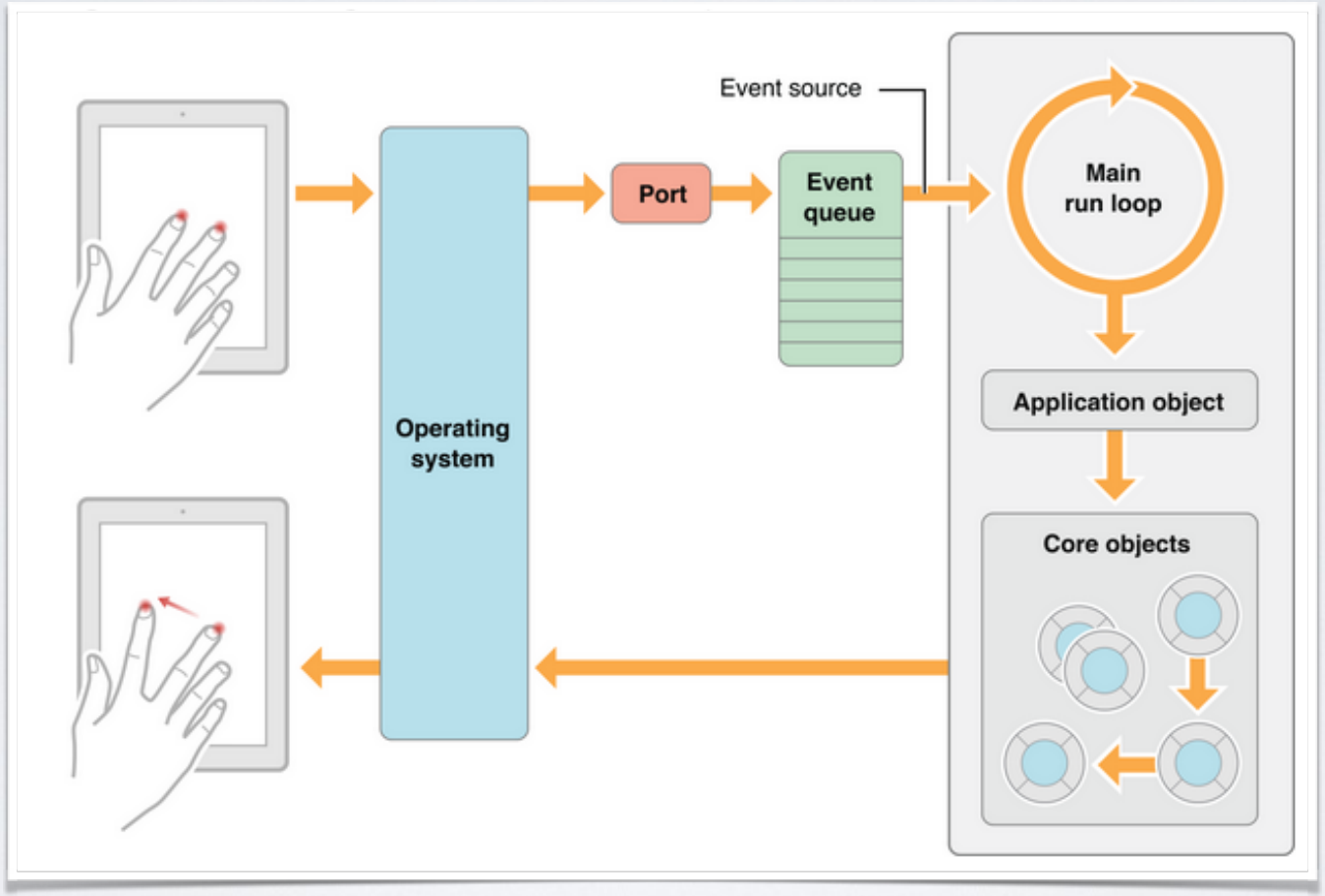
CREATE "HELLO WORLD!" LABEL



Hello World!
Demo

GroundSpeed™
rapid web + mobile software

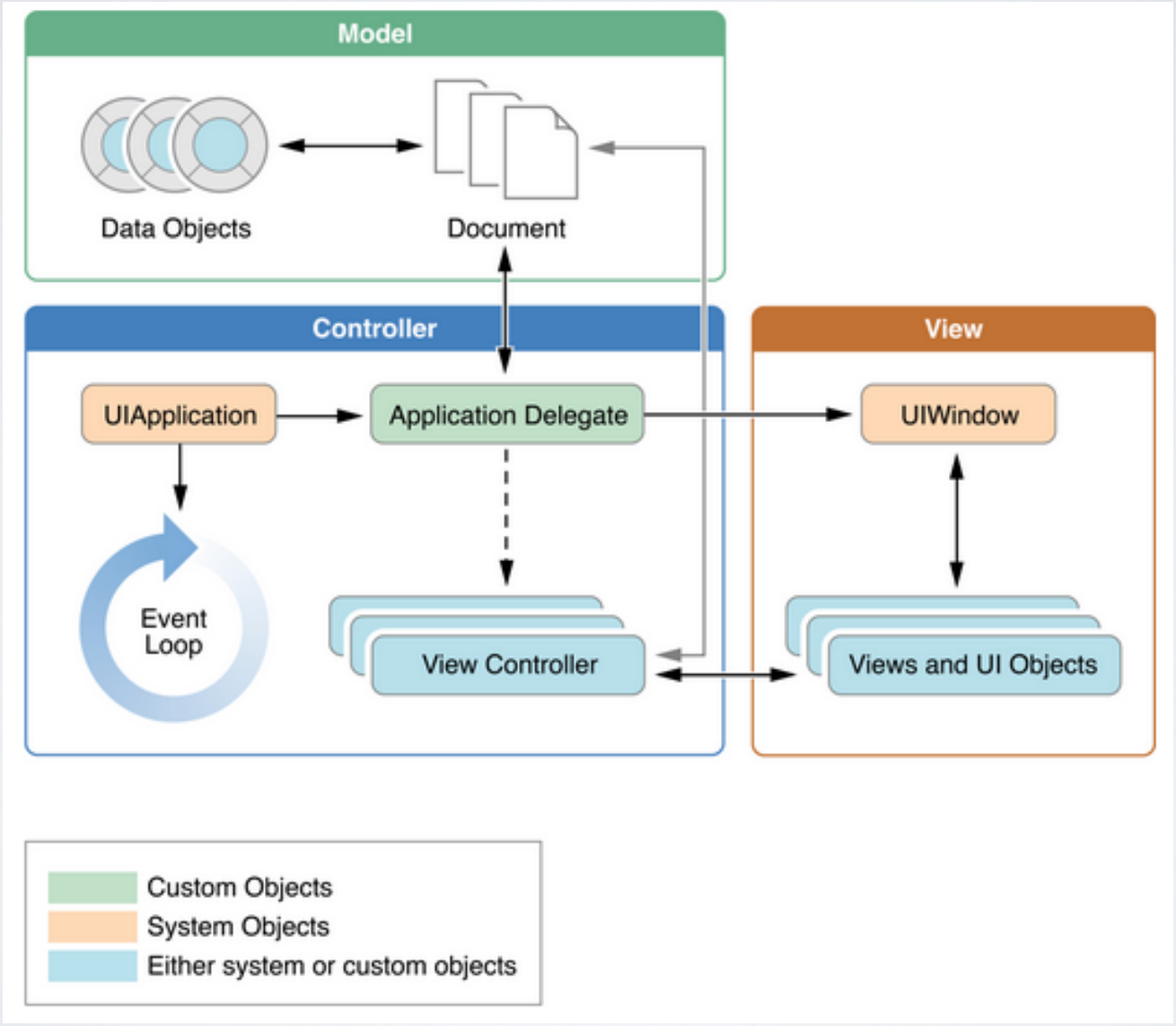
IOS APPLICATION RUNTIME CYCLE



COMMON TYPES OF EVENTS

Event type	Delivered to...	Notes
Touch	The view object in which the event occurred	Views are responder objects. Any touch events not handled by the view are forwarded down the responder chain for processing.
Remote control Shake motion events	First responder object	Remote control events are for controlling media playback and are generated by headphones and other accessories.
Accelerometer Magnetometer Gyroscope	The object you designate	Events related to the accelerometer, magnetometer, and gyroscope hardware are delivered to the object you designate.
Location	The object you designate	You register to receive location events using the Core Location framework. For more information about using Core Location, see *Location and Maps Programming Guide*.
Redraw	The view that needs the update	Redraw events do not involve an event object but are simply calls to the view to draw itself. The drawing architecture for iOS is described in *Drawing and Printing Guide for iOS*.

MODEL - VIEW - CONTROLLER



MODEL - VIEW - CONTROLLER

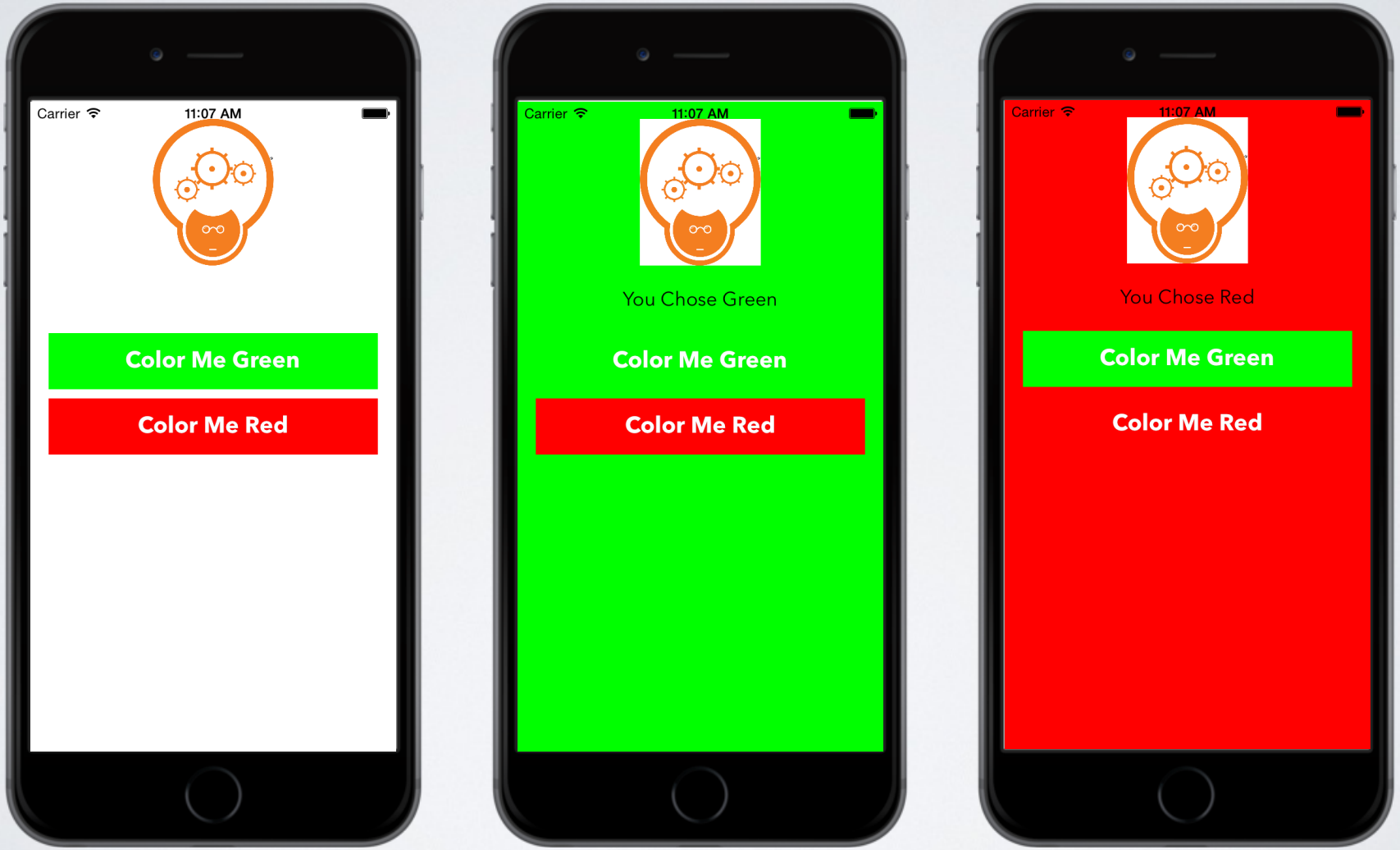
The Model: The M in MVC stands for the Model: For the Model, we will need to develop an independent class (or a set of classes) that will serve the data requirements and standard business procedures.

The View: In this context the View is the UI screen's layout (the canvas). Here, we configure many User Interface items (also known as UI controls on the screen).

The Controller: This is a class that connects the view with the model. In its simplest form, it contains the name of the UIControls (that would be used in the app), and it also includes the signatures and the detail implementations of event handlers.

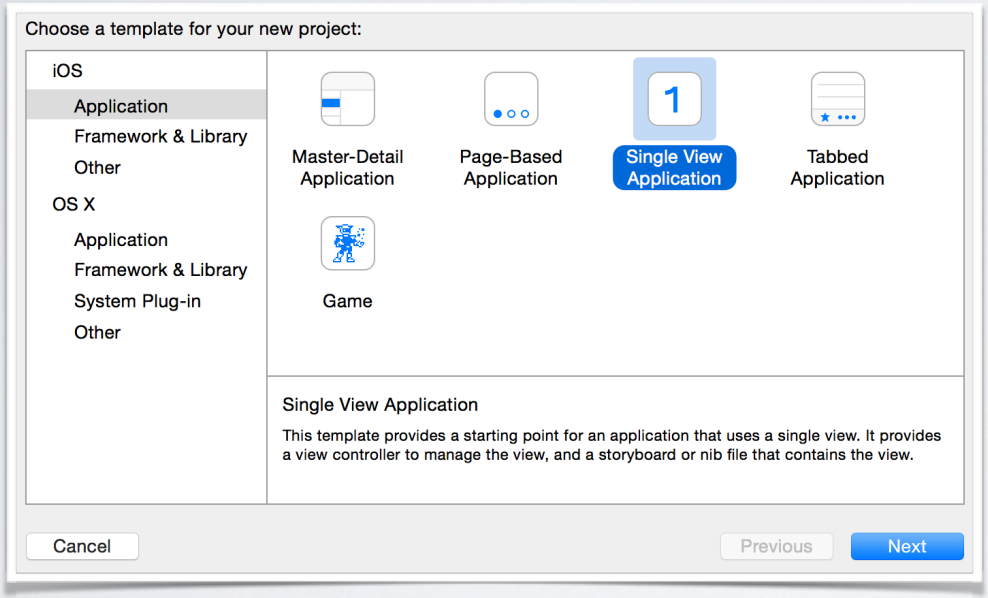
Need a Break?

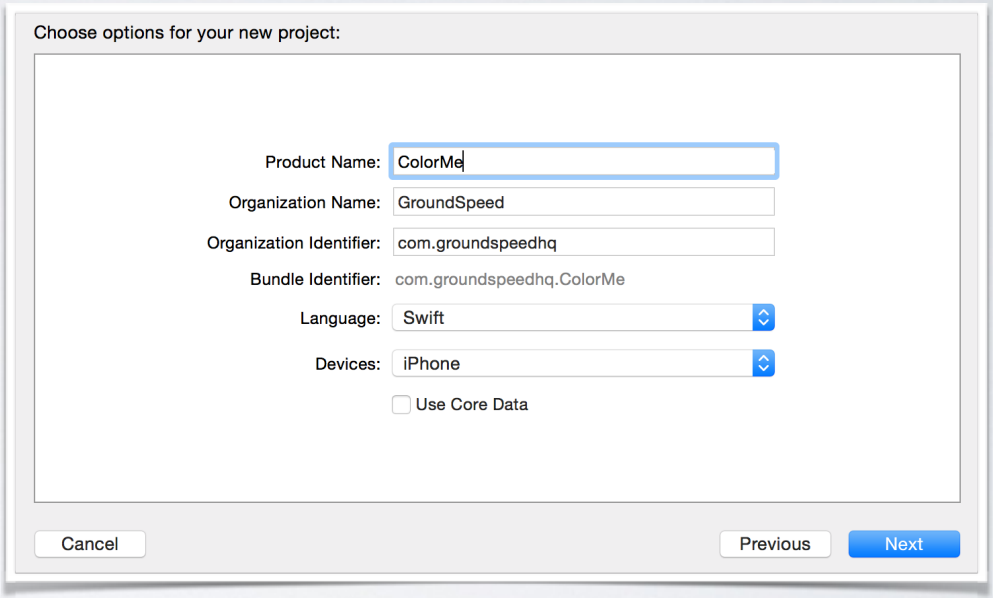
ANOTHER QUICK DEMO - [COLOR ME]



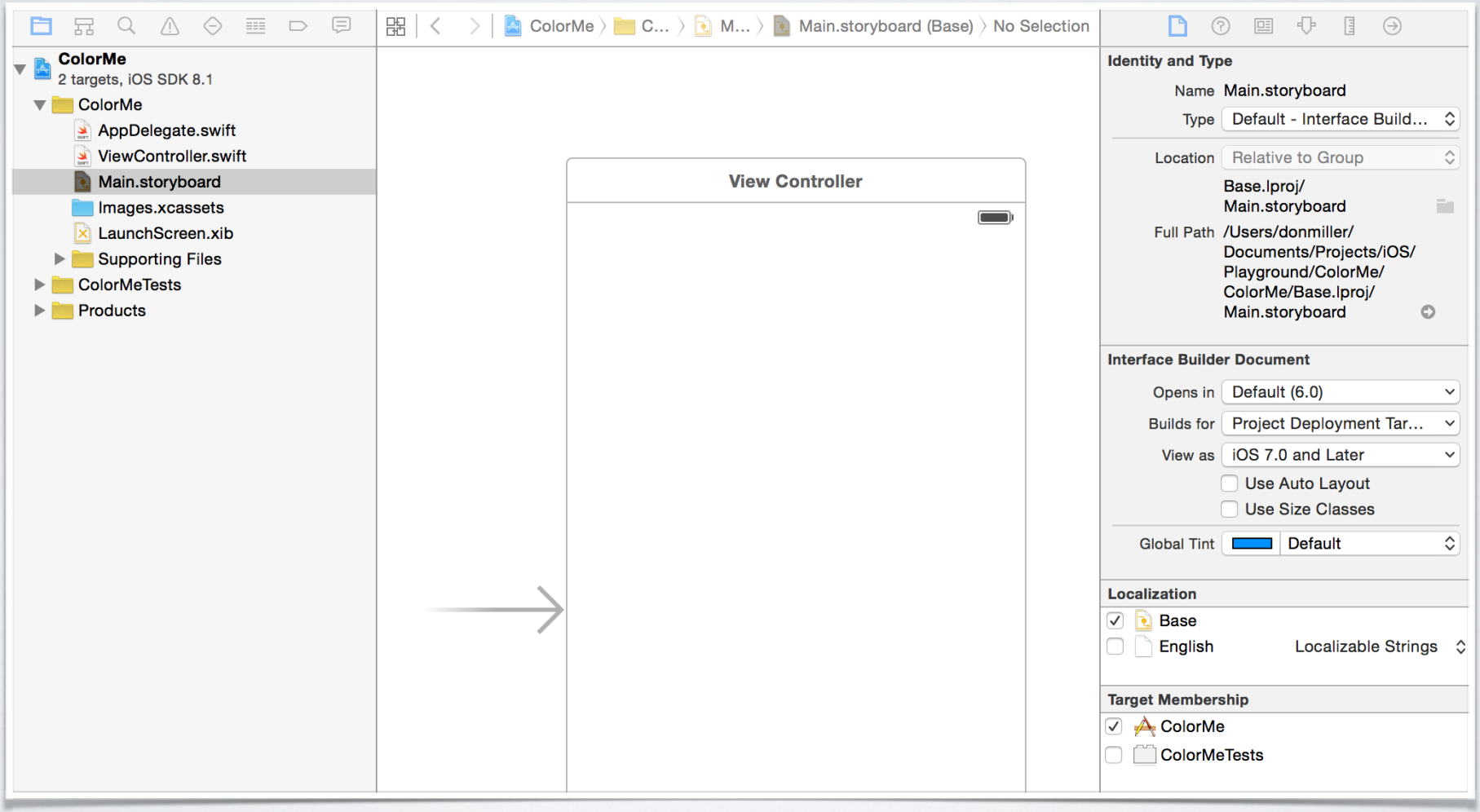
GroundSpeed™
rapid web + mobile software

CREATE A SINGLE VIEW APPLICATION



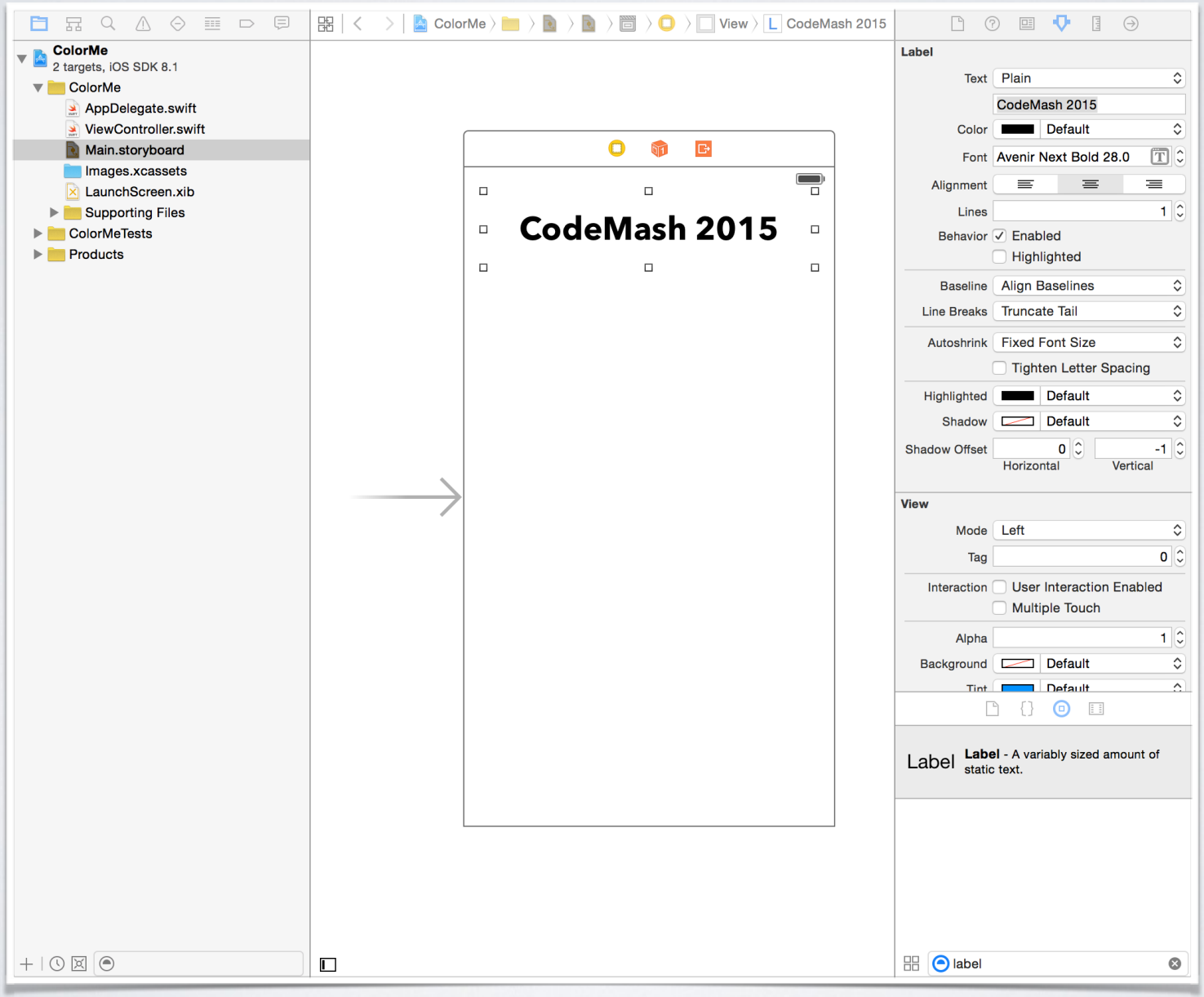


CONFIGURE STORYBOARD BY DISABLING AUTO LAYOUT AND SIZE CLASSES



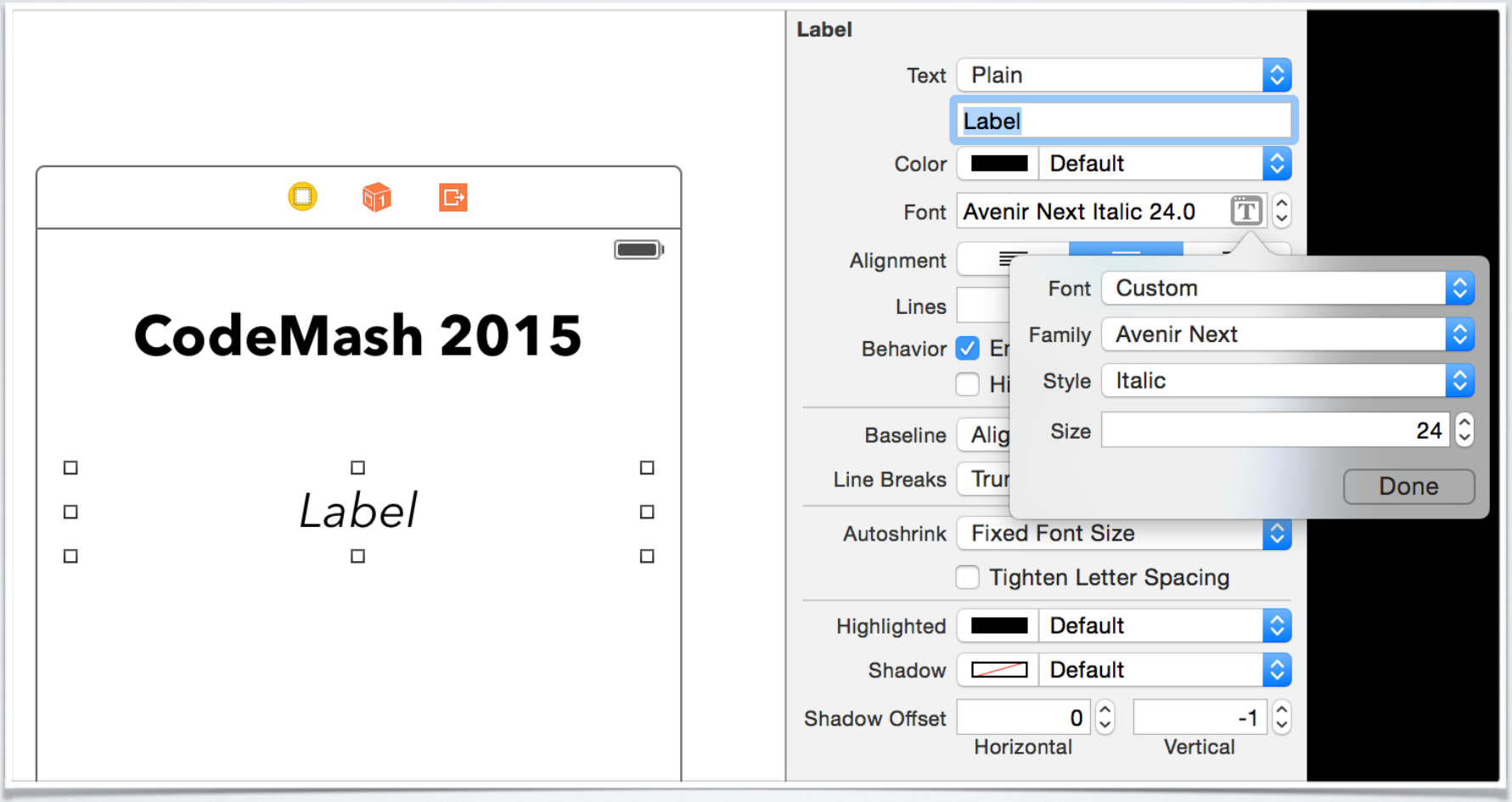
GroundSpeed™
rapid web + mobile software

DRAG A LABEL TO THE STORYBOARD

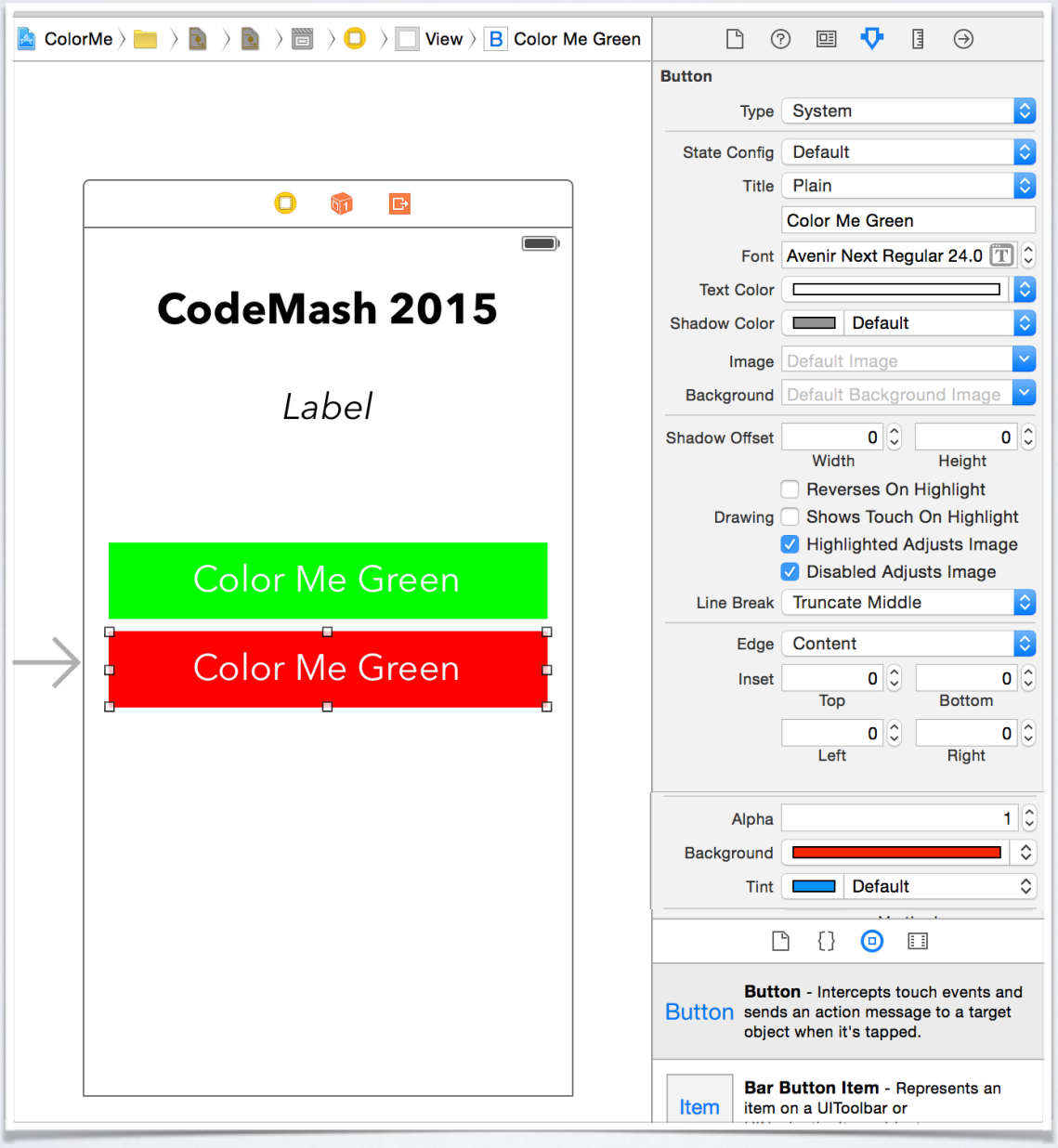


GroundSpeed™
rapid web + mobile software

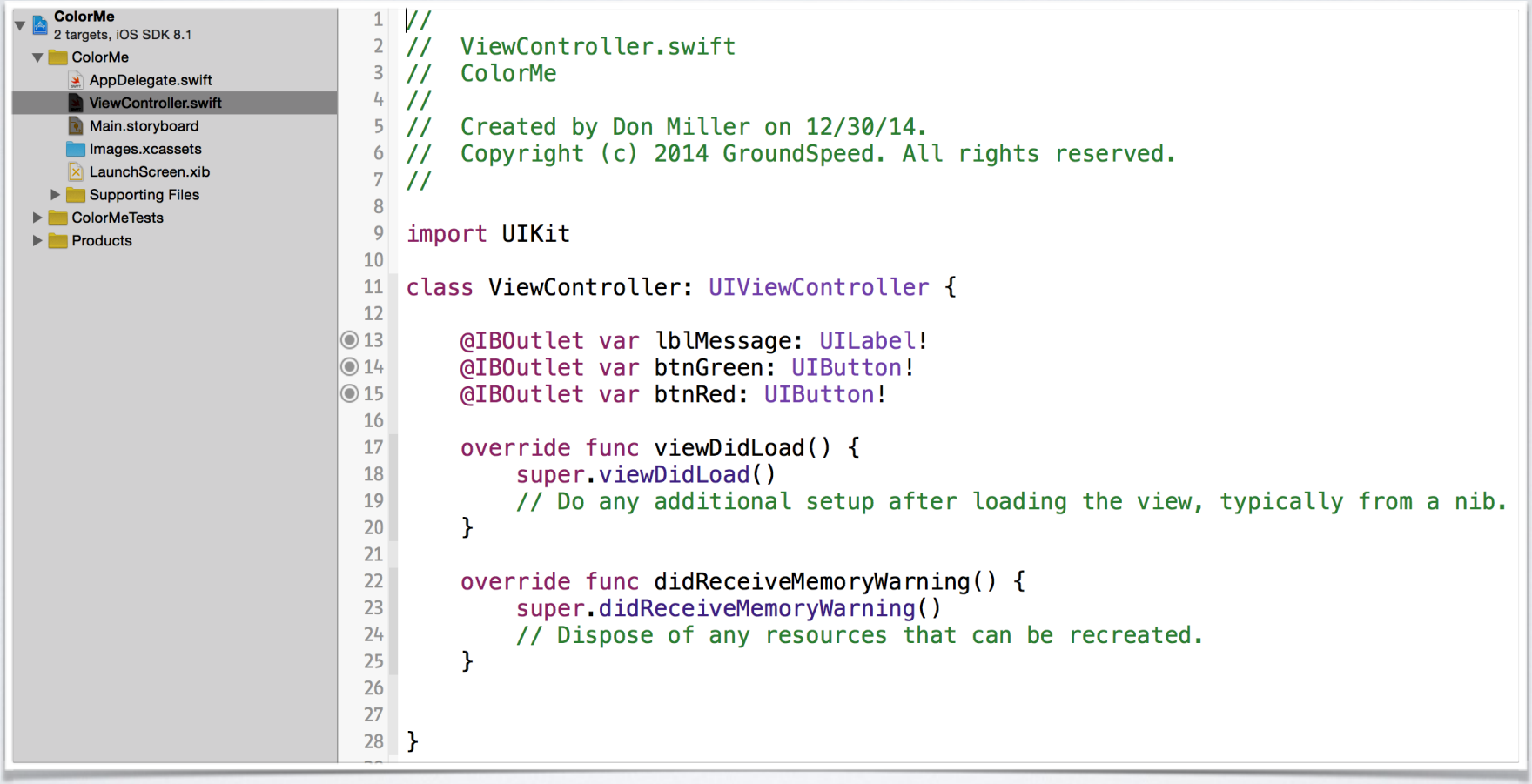
CREATE A BLANK LABEL FOR OUTPUT



CREATE TWO BUTTONS



ADD OUTLETS TO CONTROLLER

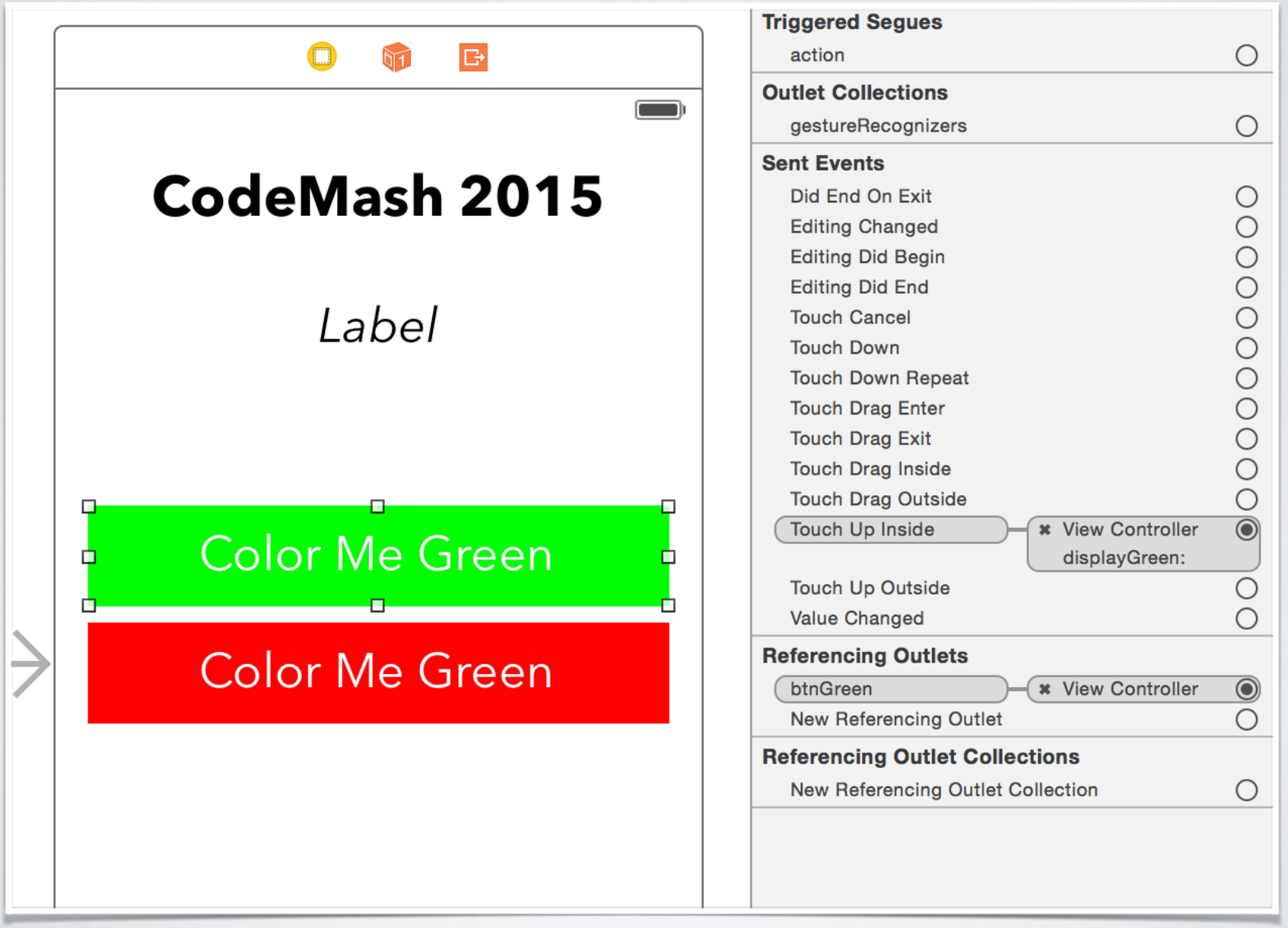



```swift
//
//  ViewController.swift
//  ColorMe
//
//  Created by Don Miller on 12/30/14.
//  Copyright (c) 2014 GroundSpeed. All rights reserved.
//

import UIKit

class ViewController: UIViewController {

    @IBOutlet var lblMessage: UILabel!
    @IBOutlet var btnGreen: UIButton!
    @IBOutlet var btnRed: UIButton!

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }


}
```

ADD ACTIONS TO CONTROLLER AND UPDATE VIEWDIDLOAD


```
17      override func viewDidLoad() {
18          super.viewDidLoad()
19          // Do any additional setup after loading the view, typically from a nib.
20          lblMessage.text = ""
21      }
22
23      override func didReceiveMemoryWarning() {
24          super.didReceiveMemoryWarning()
25          // Dispose of any resources that can be recreated.
26      }
27
28      @IBAction func displayGreen(sender: AnyObject) {
29          lblMessage.text = "You wanted green"
30          self.view.backgroundColor = UIColor.greenColor()
31      }
32
33      @IBAction func displayRed(sender: AnyObject) {
34          lblMessage.text = "You wanted red"
35          self.view.backgroundColor = UIColor.redColor()
36      }
37  }
```

CONNECTING THE OUTLETS AND ACTIONS



Color Me Demo

GroundSpeed™
rapid web + mobile software

SWIFT BASICS

Xcode Playgrounds

Variables & Constants (Mutable and Immutable)

Types

- Strings, Integers, Floats, Booleans

Operators

- Binary and Unary Operators

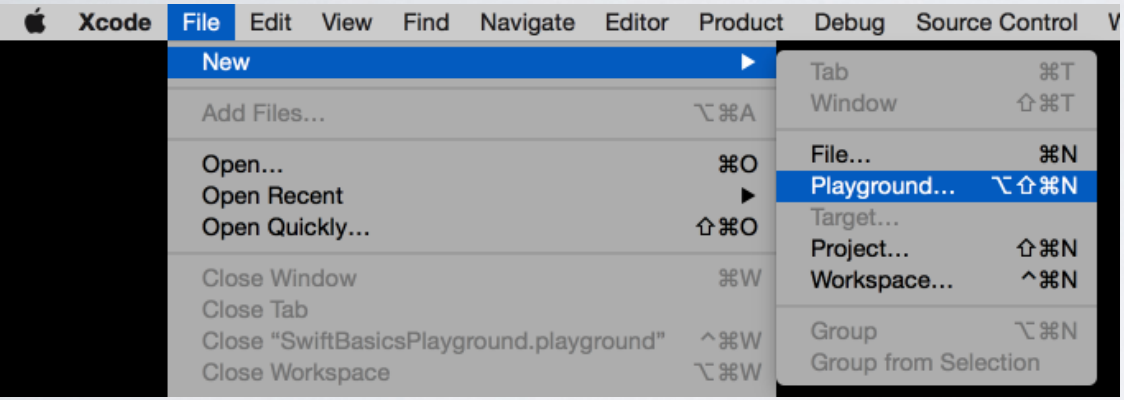
Collection Types

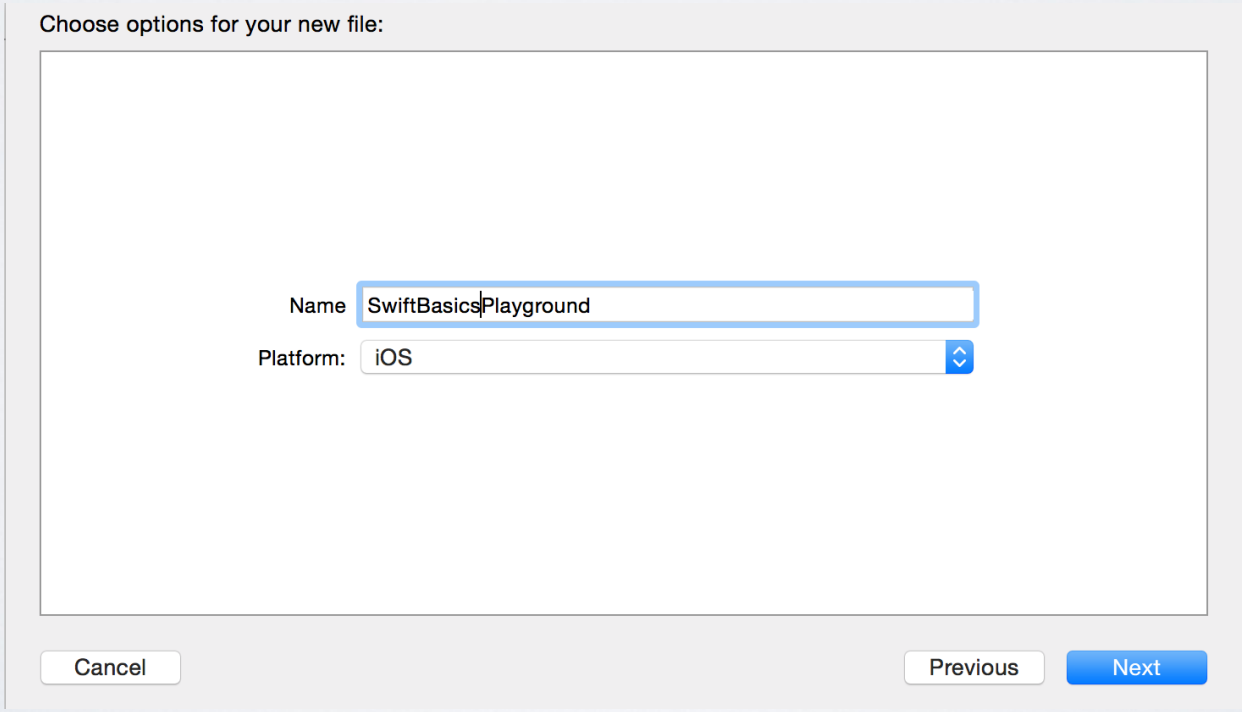
- Arrays and Dictionaries

Control Flow

- For-In, While, For-Condition, If, Switch, Comparison and Logical Operators

EXAMPLE: HAVE FUN IN THE PLAYGROUND





GroundSpeed™
rapid web + mobile software

ABOUT MUTABLE AND IMMUTABLE OBJECTS OR VARIABLES AND CONSTANTS

```
3  import UIKit
4
5  let immutable = "This cannot be changed"
6  immutable = "This will give an error"
7
8  var mutable = "This cannot be changed"
9  mutable = "This will NOT give an error"
10
```

"This cannot be changed"

"This cannot be changed"
"This will NOT give an error"

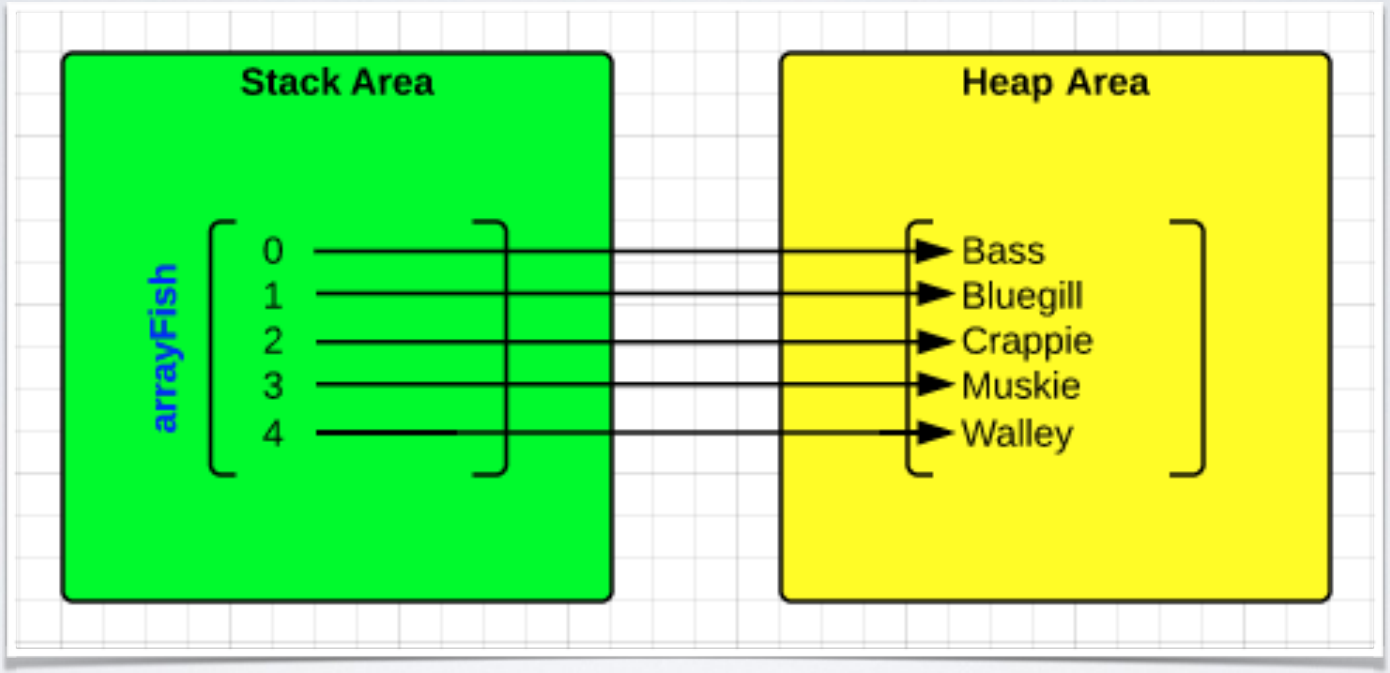
DECLARING TYPES: STRINGS, INTEGERS, FLOATS, AND BOOLEANS

```
5  let string1 = "This is a string"                    "This is a string"
6  let string2: String = "This is string typed"        "This is string typed"
7
8  var intNumber1 = 1                                  1
9  intNumber1 = 2.5
10     Type 'Int' does not conform to protocol 'FloatLiteralConvertible'
11 let intNumber2: Int = 3                             3
12
13 var floatNumber1 = 1.0                              1.0
14 floatNumber1 = 2                                    2.0
15 floatNumber1 = 2.5                                  2.5
16
17 let floatNumber2: Float = 1                         1.0
18
19 let bool1 = true                                     true
20 let bool2: Bool = false                              false
```

OPERATORS: BINARY AND UNARY
...AND MAYBE A TERNARY TOO

```
3  import UIKit                          
4                                        
5  // binary                            
6  var a = 1                             1
7  a = a + 1                             2
8  a = a - 1                             1
9  a = a / 1                             1
10 a = a * 4                             4
11 a = a % 3                             1
12                                       
13 // unary                             
14 var b = 1                             1
15 ++b                                   2
16 b++                                   2
17 b                                     3
18 b--                                   3
19 b                                     2
20                                       
21 // ternary                           
22 var bool = true                       true
23 var c = (bool ? 1 : 3)                1
24 bool = false                          false
25 c                                     1
26 c = (bool ? 1 : 3)                    3
```

ARRAYS




```
5  var arrayFish = ["Bass", "Bluegill",
       "Crappie", "Muskie", "Walley"]
6
7  arrayFish.count                          5
8  arrayFish[3]                             "Muskie"
9  arrayFish[5]   Execution was interrupted, reason: EXC_...   error
```

["Bass", "Bluegill", "Crappie", "Muskie", "Walley"]

CREATING ARRAYS AND DICTIONARIES WITH SWIFT

```swift
// Arrays
var arrayFish = ["Bass", "Bluegill", "Crappie", "Muskie", "Walley"]

arrayFish.count
arrayFish[3]


// Dictionaries
var dictStates = ["OH": "Ohio", "MI": "Michigan", "KY": "Kentucky", "IN":
    "Indiana", "IL": "Illinois", "PA": "Pennsylvania", "NY": "New York"]

dictStates.count
dictStates["OH"]!

// For Loop through dictionary
for (stateCode, stateName) in dictStates {
    println("\(stateCode): \(stateName)")
}

for stateCode in dictStates.keys {
    println("\(stateCode)")
}

for stateName in dictStates.values {
    println("\(stateName)")
}

// Array of stateCodes
let stateCodes = [String](dictStates.keys)
```

CONTROL FLOW - LOOPING

```swift
// For-In
for index in 1...5 {
    println("\(index) times 5 is \(index * 5)")
}

let arrayFish = ["Bass", "Bluegill", "Crappie", "Muskie", "Walley"]
for fish in arrayFish {
    println("Fish array: \(fish)")
}

// For
for var index = 0; index < 5; ++index {
    println("For Index is \(index)")
}

// While
var index = 0
var max = 5
while index < max {
    println("While Index is \(index++)")
}

// Do-While
index = 0
max = 5
do {
    println("Do-While Index is \(index++)")
} while index < max
```

Console Output

```
1 times 5 is 5
2 times 5 is 10
3 times 5 is 15
4 times 5 is 20
5 times 5 is 25
Fish array: Bass
Fish array: Bluegill
Fish array: Crappie
Fish array: Muskie
Fish array: Walley
For Index is 0
For Index is 1
For Index is 2
For Index is 3
For Index is 4
While Index is 0
While Index is 1
While Index is 2
While Index is 3
While Index is 4
Do-While Index is 0
Do-While Index is 1
Do-While Index is 2
Do-While Index is 3
Do-While Index is 4
```

CONTROL FLOW - DECISION

```
// IF
var temperatureInFahrenheit = 90
if temperatureInFahrenheit <= 32 {
    println("It's very cold. Consider wearing a scarf.")
} else if temperatureInFahrenheit >= 86 {
    println("It's really warm. Don't forget to wear sunscreen.")
} else {
    println("It's not that cold. Wear a t-shirt.")
}

// SWITCH
let someCharacter: Character = "e"
switch someCharacter {
case "a", "e", "i", "o", "u":
    println("\(someCharacter) is a vowel")
case "b", "c", "d", "f", "g", "h", "j", "k", "l", "m",
"n", "p", "q", "r", "s", "t", "v", "w", "x", "y", "z":
    println("\(someCharacter) is a consonant")
default:
    println("\(someCharacter) is not a vowel or a consonant")
}
```

```
90

"It's really warm. Don't forget to wear sunscreen."

"e"

"e is a vowel"
```

GroundSpeed™
rapid web + mobile software

CLASS ILLUSTRATIONS

```
5  class Person {
6      var ssn: String!
7      var name: String!
8      var city: String!
9      var state: String!
10 }
11
12 class Employee: Person {
13     var empId: String!
14     var department: String!
15     var healthInsurance: String!
16
17     func printChecks() {
18         println("Printing Checks")
19     }
20 }
21
22 class SalariedEmployee: Employee {
23     var salary: String!
24 }
25
26 class HourlyEmployee: Employee {
27     var hourlyRate: String!
28 }
29
```

```
30 let hourlyEmployee = HourlyEmployee()
31 hourlyEmployee.name = "Don Miller"
32 hourlyEmployee.empId = "1234"
33 hourlyEmployee.hourlyRate = "9.99"
34
35 var output = "Name: \(hourlyEmployee.name) \r"
36 output += "Employee ID is \(hourlyEmployee.empId) \r"
37 output += "Hourly wage is \(hourlyEmployee.hourlyRate) \r"
38
39 println("\(output)")
40 hourlyEmployee.printChecks()
```

Console Output

```
Name: Don Miller
Employee ID is 1234
Hourly wage is 9.99
Printing Checks
```

STORYBOARD OVERVIEW

Storyboards are graphic organizers displayed in sequence for the purpose of pre-visualizing a motion picture or animation. The storyboarding process, in the form it is known today, is used by developers to give the designer, developer, and end user an opportunity to see what the application or web site will look like at a high level.

In iOS, a Storyboard is a container of screens that may have embedded view controllers, navigation controllers, and tab bar controllers. It allows the user to graphically connect screens to one another through embedded objects on the screen. These connections are called segues (pronounced seg-wey). The storyboard eliminates the need for xib files and has one file that contains all of your screens and transitions. It is possible to connect two storyboard files with one another; however, unlikely with smaller applications.

PROS OF STORYBOARDS

Pros of using Storyboards:

- With a storyboard you have a better conceptual overview of all the screens in your app and the connections between them. It's easier to keep track of everything because the entire design is in a single file, rather than spread out over many separate nibs.

- The storyboard describes the transitions between the various screens. These transitions are called "segues" and you create them by simply ctrl-dragging from one view controller to the next. Thanks to segues you need less code to take care of your UI.

- Storyboards make working with table views a lot easier with the new prototype cells and static cells features. You can design your table views almost completely in the storyboard editor, something else that cuts down on the amount of code you have to write.

CONS OF STORYBOARDS

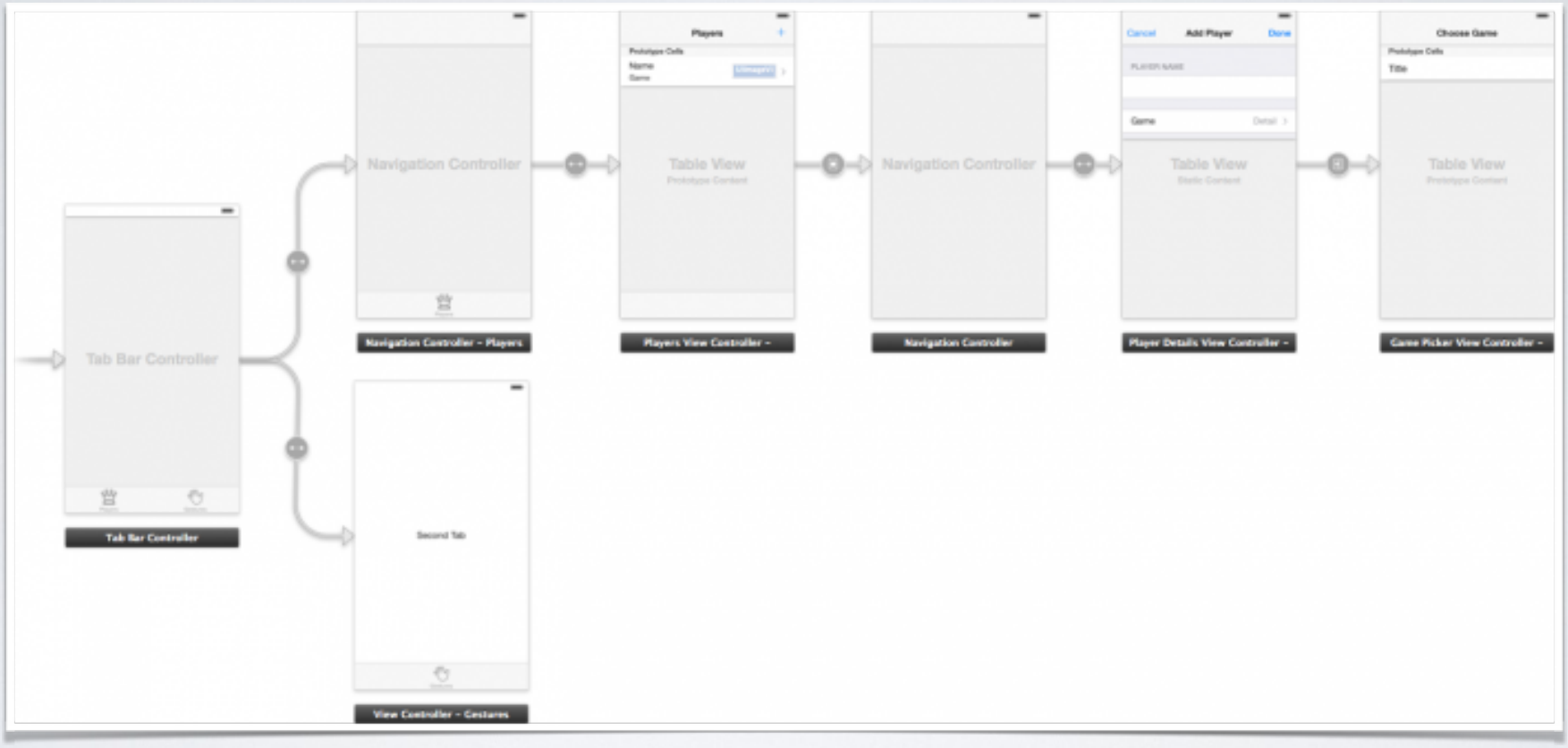
Cons of using Storyboards:

- It's not easy to work with storyboards in a team, since only one participant can work on the storyboard at once (because it's one binary file).

- If you need to do things storyboard doesn't offer, it's not quite easy to get storyboard mixed with programmatically created views)

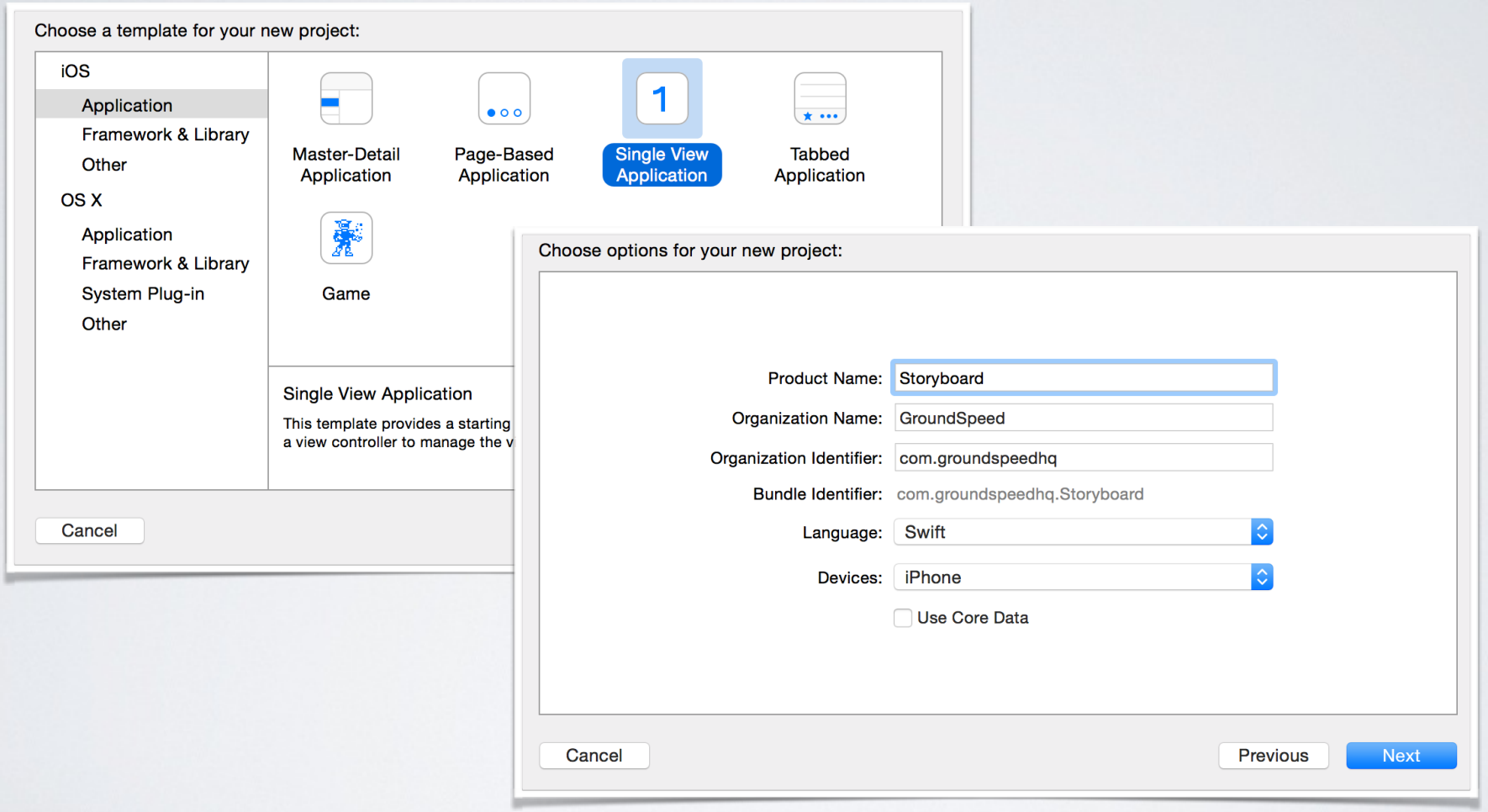
- Since all objects are in one file, a larger project could cause the storyboard to be very slow and unresponsive. It loads the entire thing into memory. It loads the entire item into memory.

STORYBOARD OVERVIEW



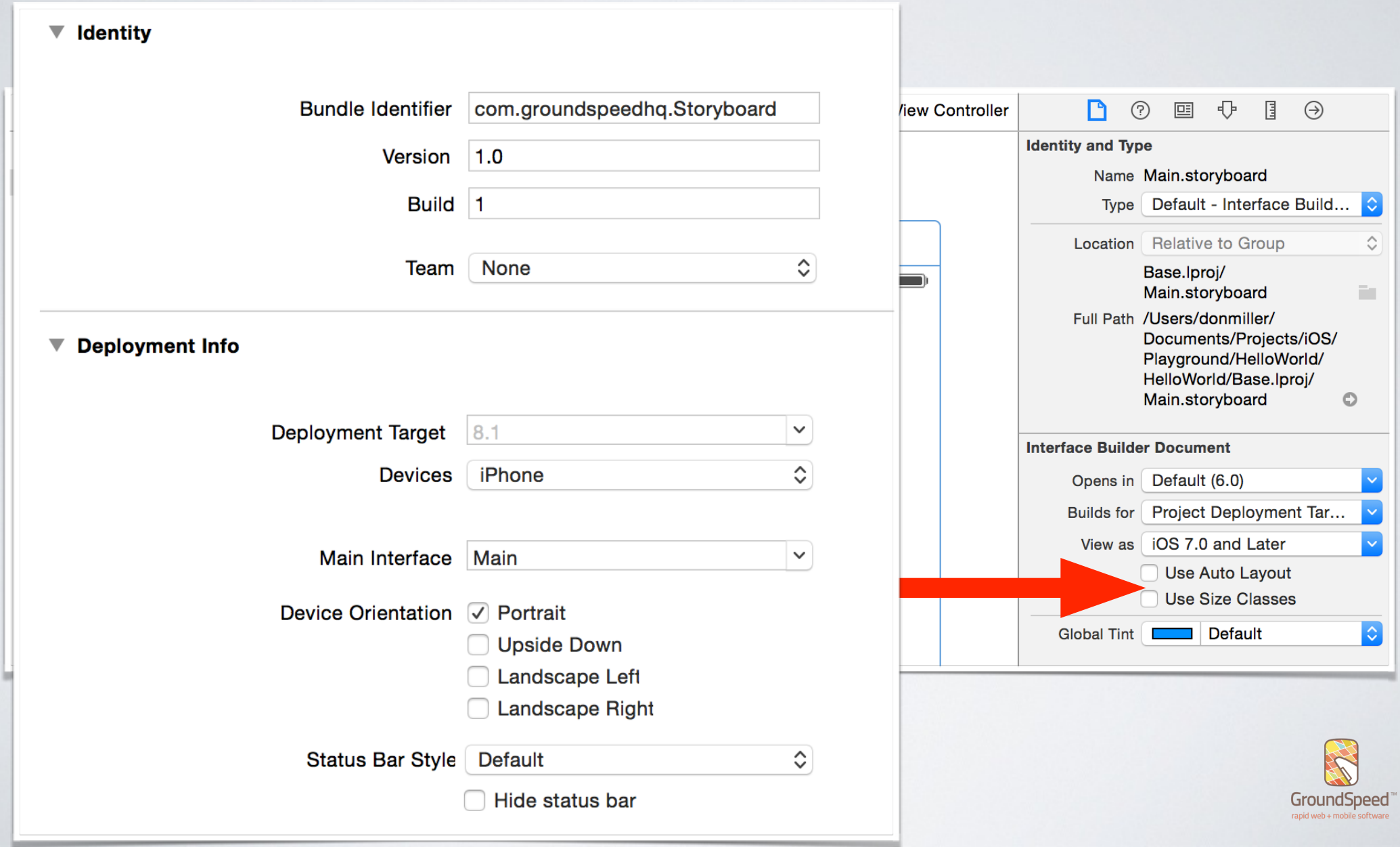
GroundSpeed™
rapid web + mobile software

STORYBOARD DEMO
STEP 1: CREATE A NEW SINGLE VIEW APPLICATION



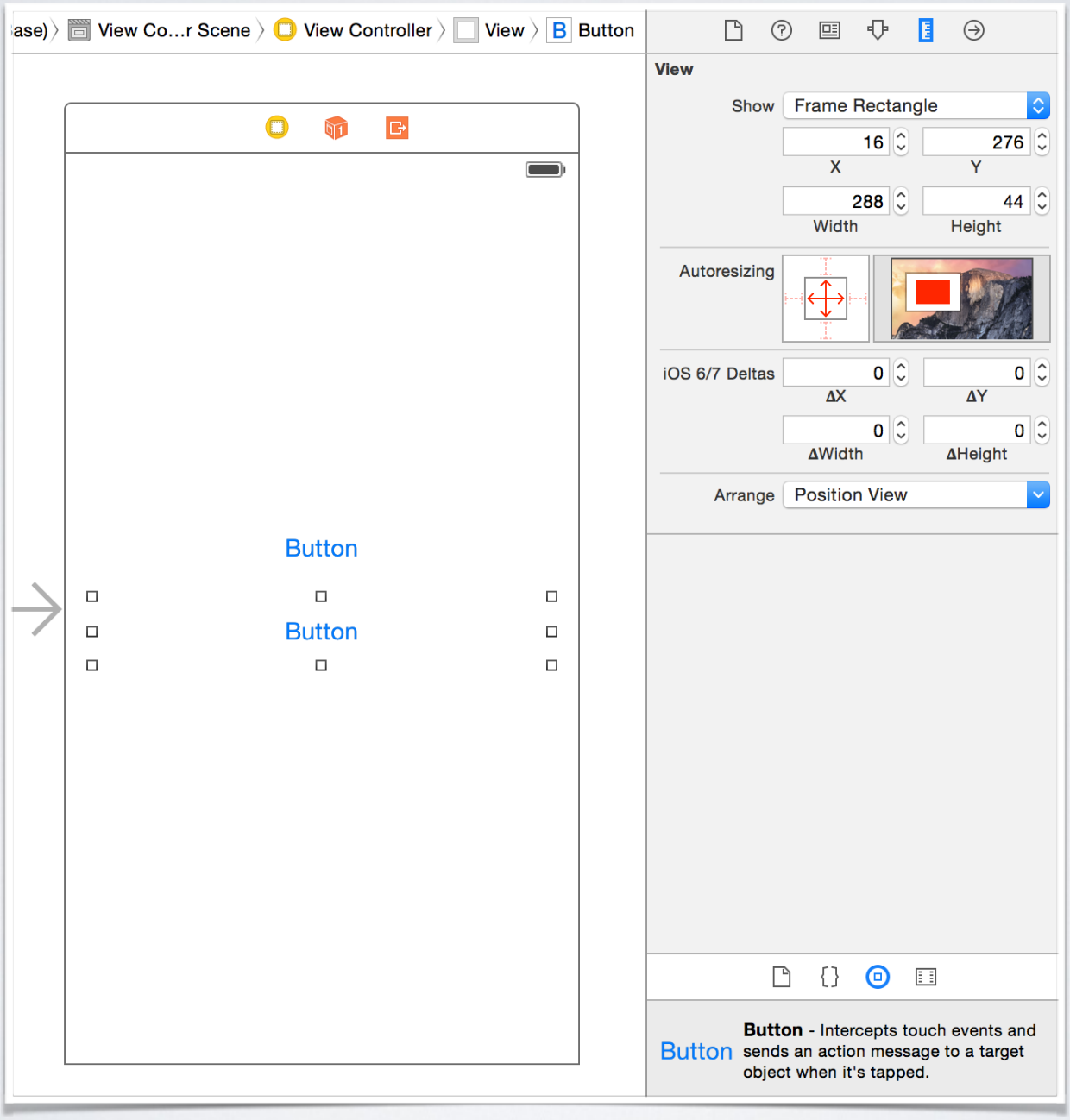
GroundSpeed™
rapid web + mobile software

STEP 2: OBSERVE THE SUMMARY PAGE AND TURN OFF SIZE CLASSES



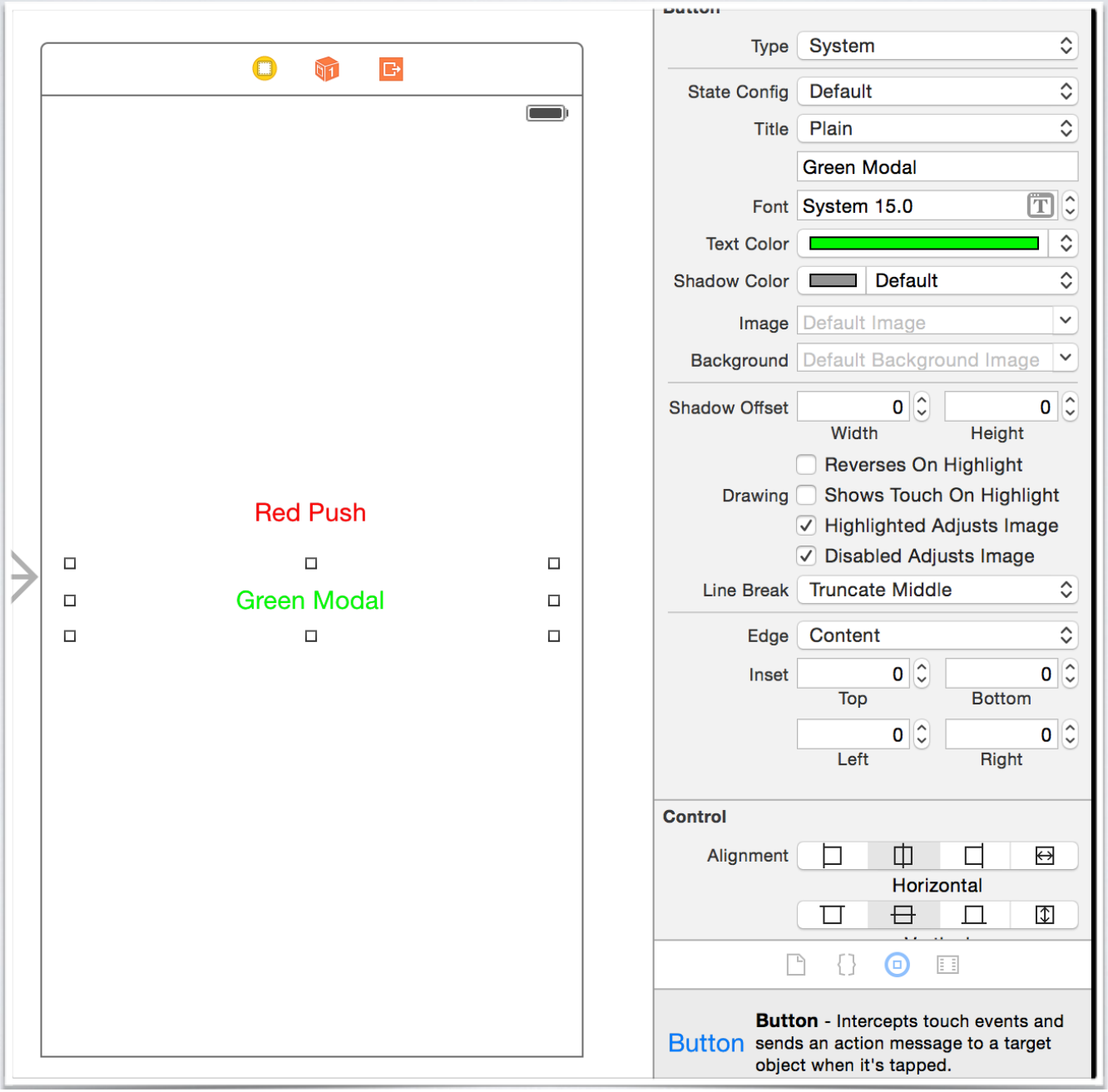
GroundSpeed™
rapid web + mobile software

STEP 3: ADD 2 BUTTONS TO THE VIEW CONTROLLER
STEP 4: LET'S MAKE THEM THE SAME WIDTH IN SIZE INSPECTOR



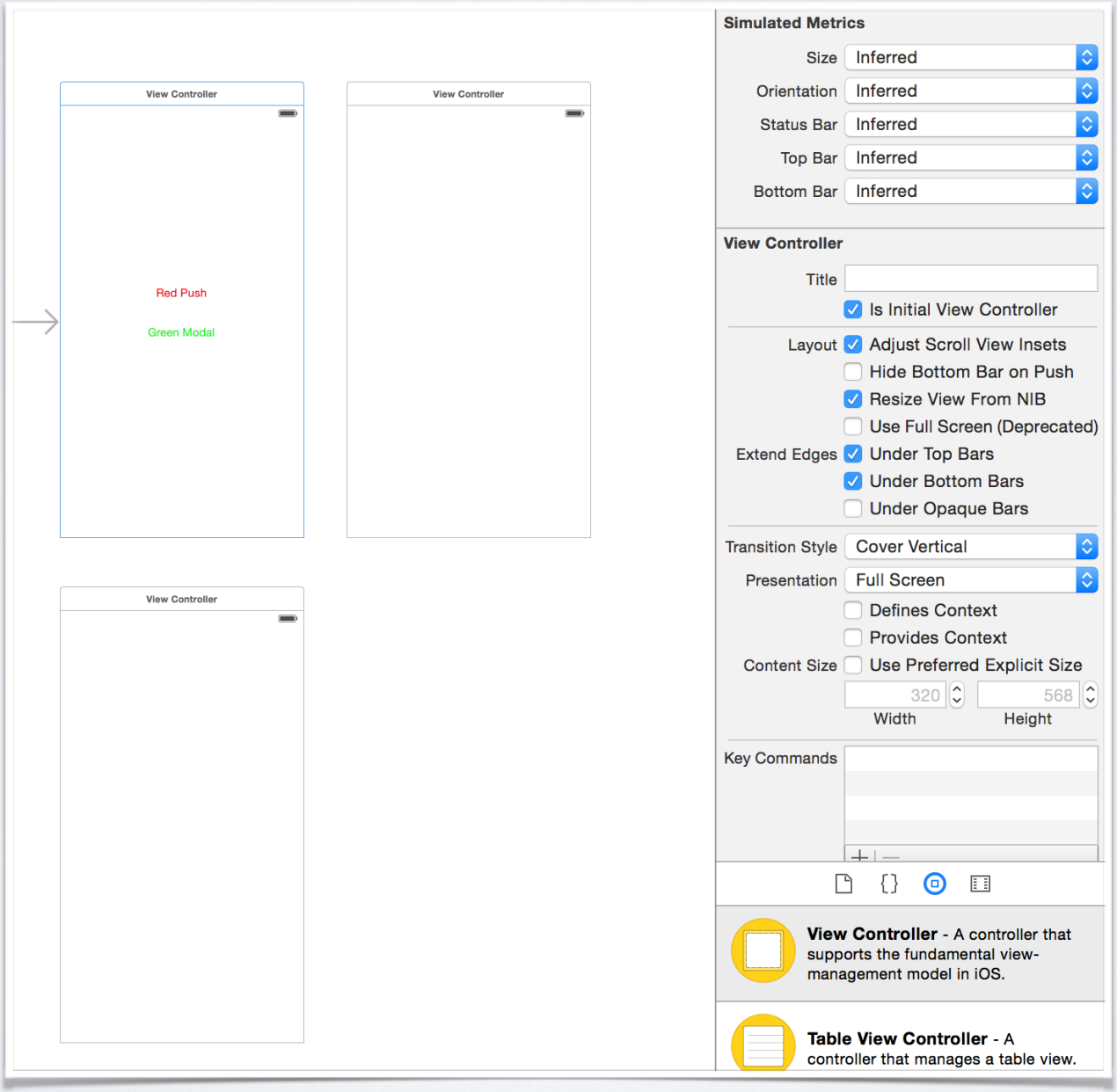
GroundSpeed™
rapid web + mobile software

STEP 5: CHANGE THE BUTTON LABELS TEXT TO RED AND GREEN

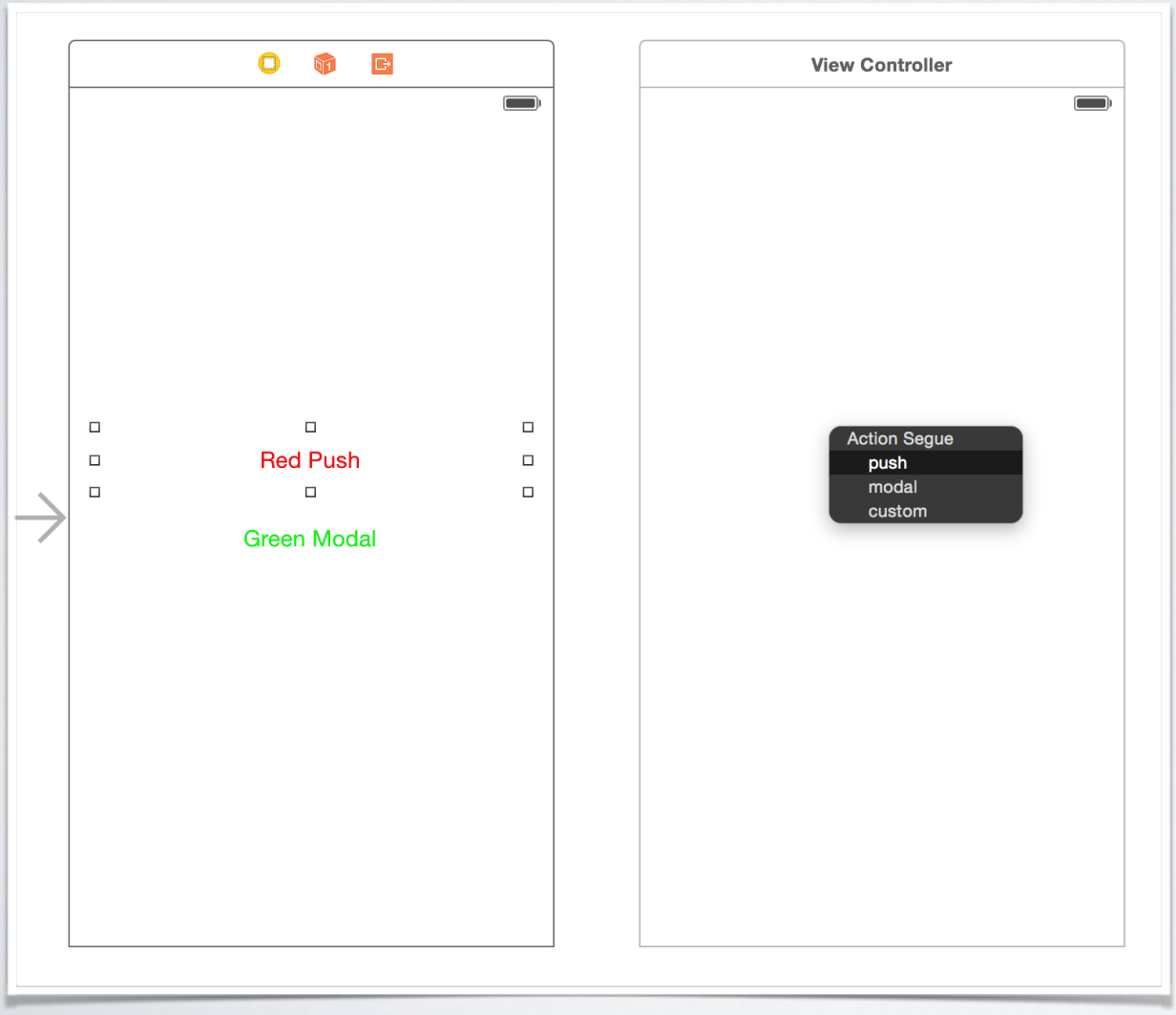


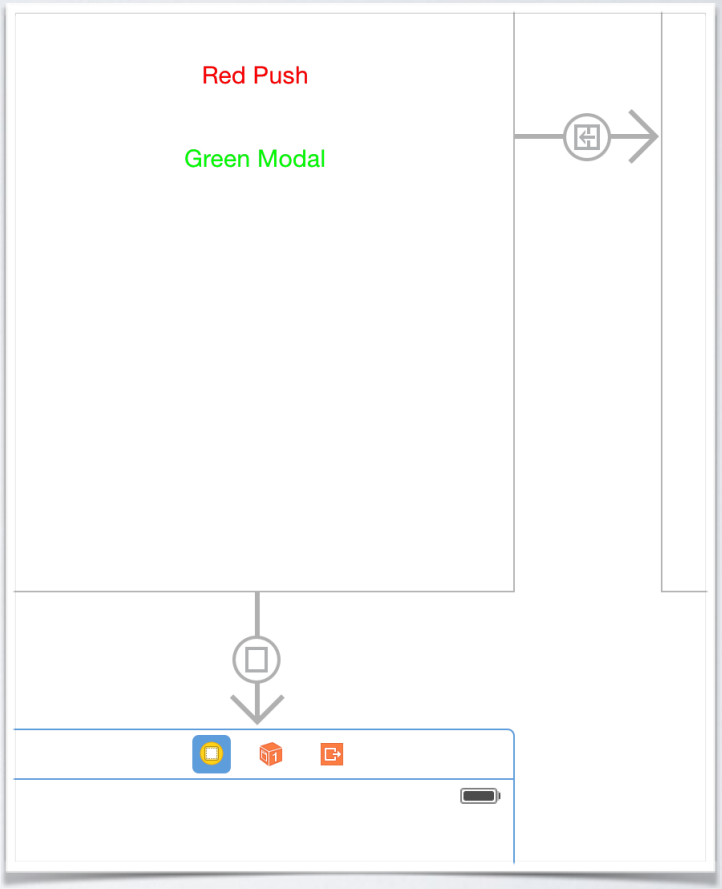
GroundSpeed™
rapid web+mobile software

STEP 6: ADD TWO VIEW CONTROLLERS TO THE STORYBOARD BY DRAGGING THEM OVER



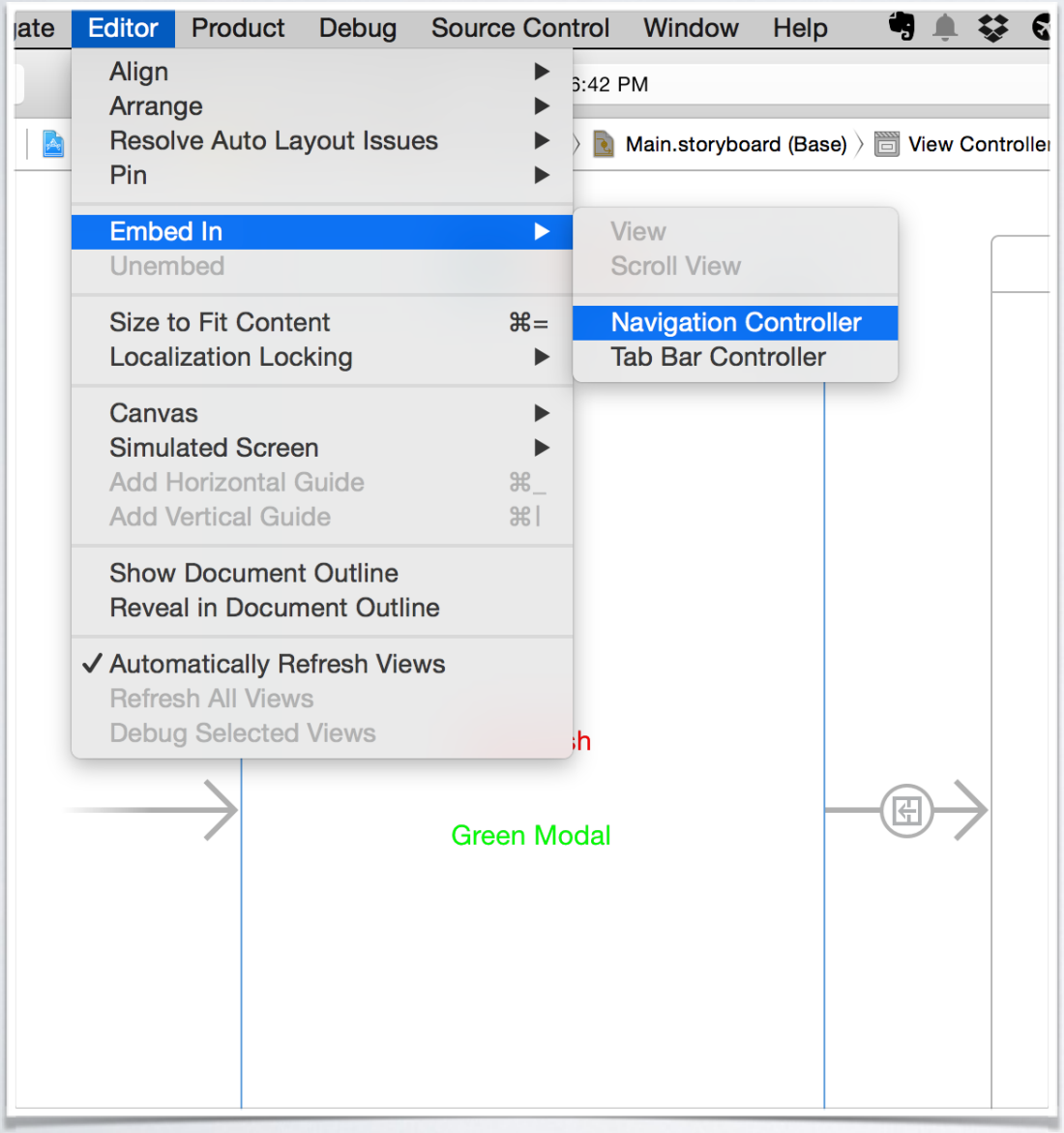
STEP 7: CONNECT THE BUTTONS TO THE NEW VIEW CONTROLLERS BY HOLDING DOWN THE [CONTROL] KEY AND DRAGGING FROM THE BUTTON TO THE NEW VIEW CONTROLLER

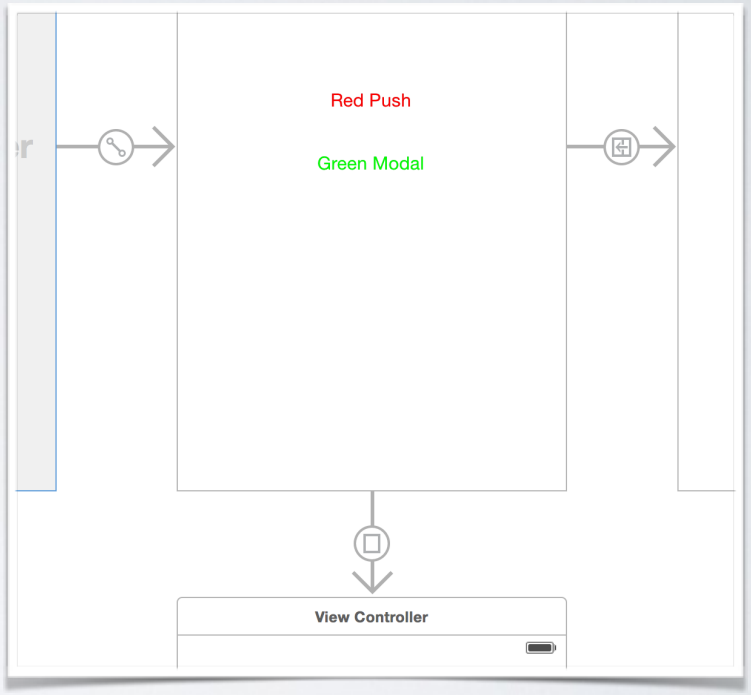




GroundSpeed™
rapid web + mobile software

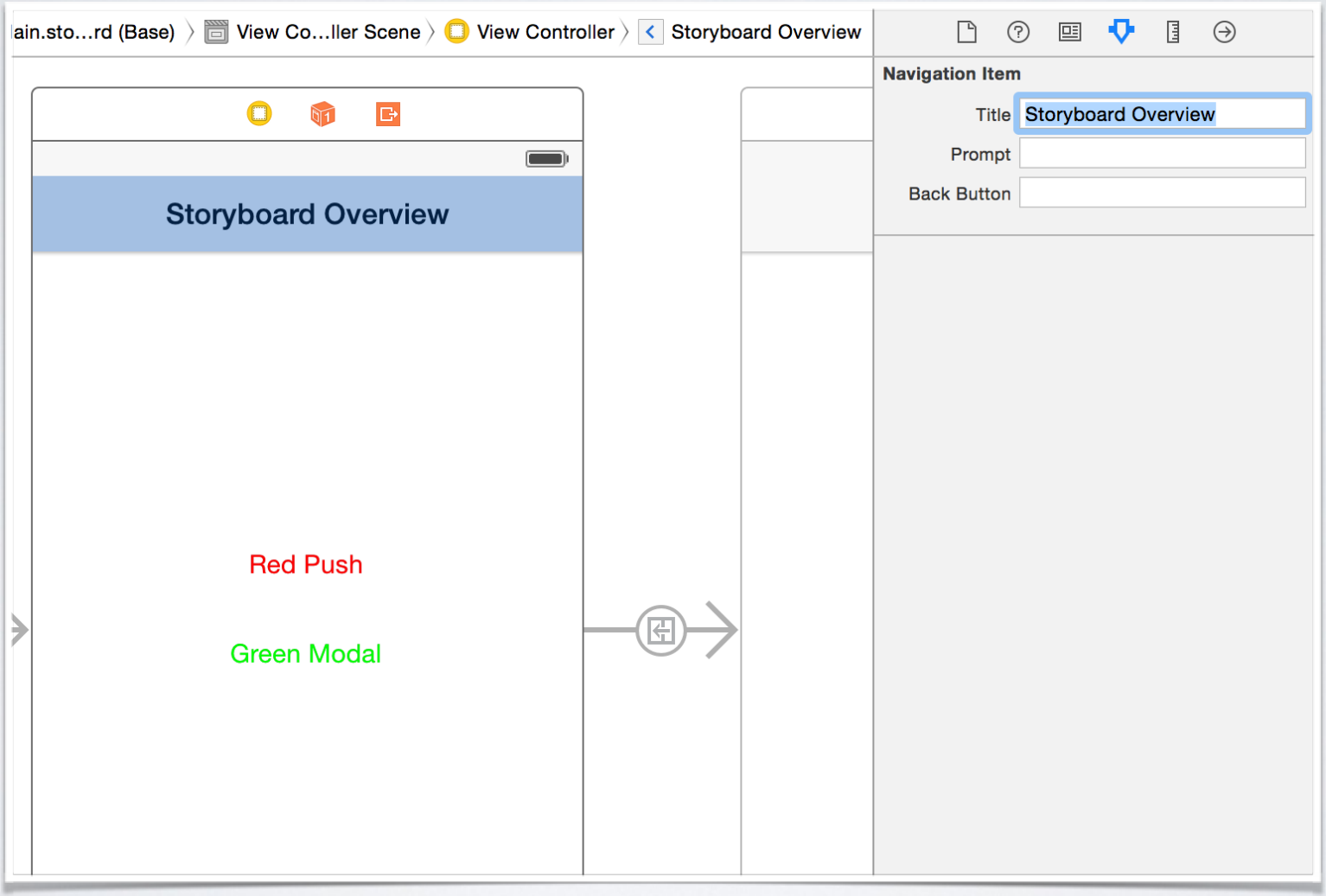
STEP 8: EMBED A NAVIGATION CONTROLLER ON YOUR PRIMARY VIEW



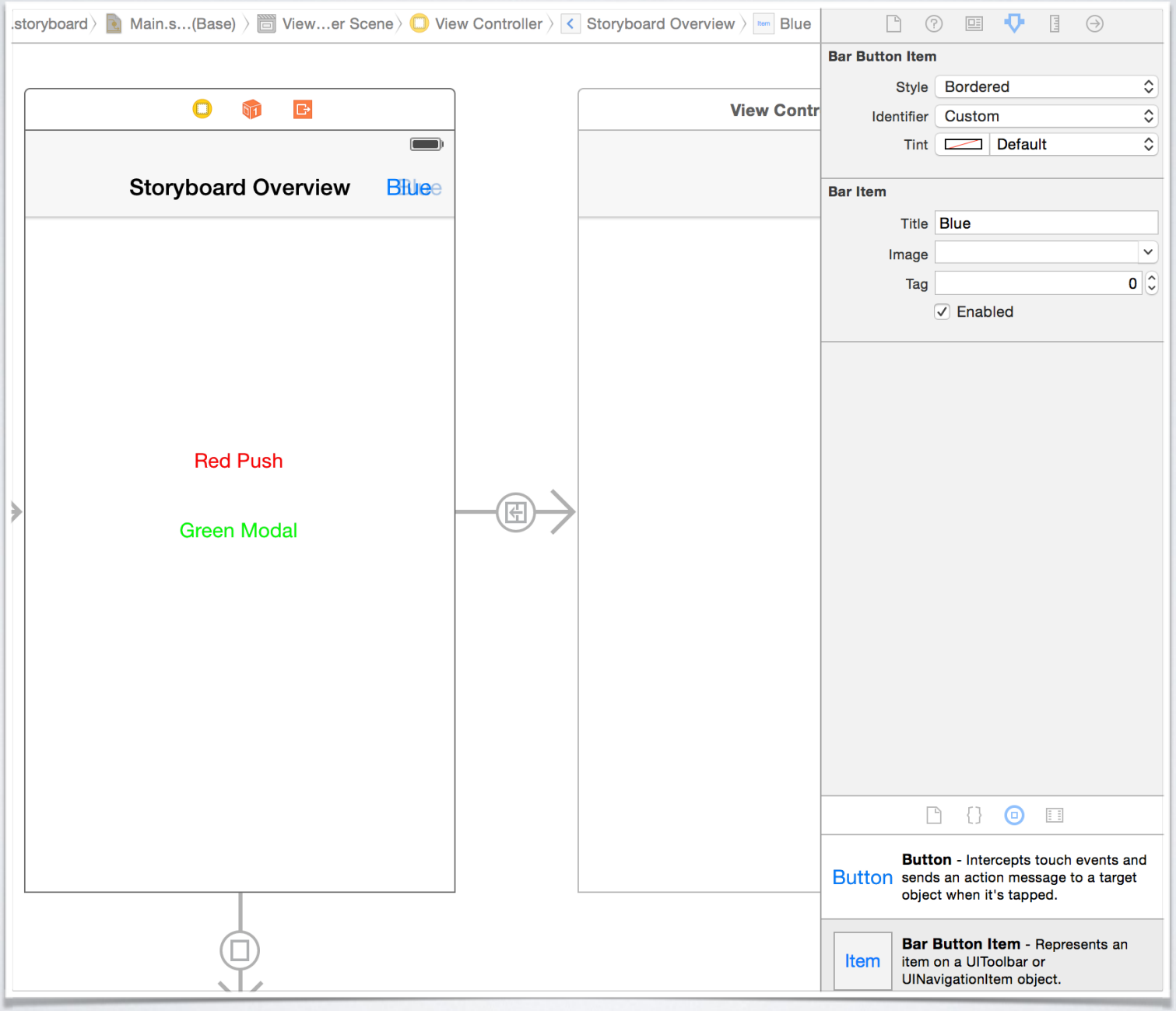


GroundSpeed™
rapid web + mobile software

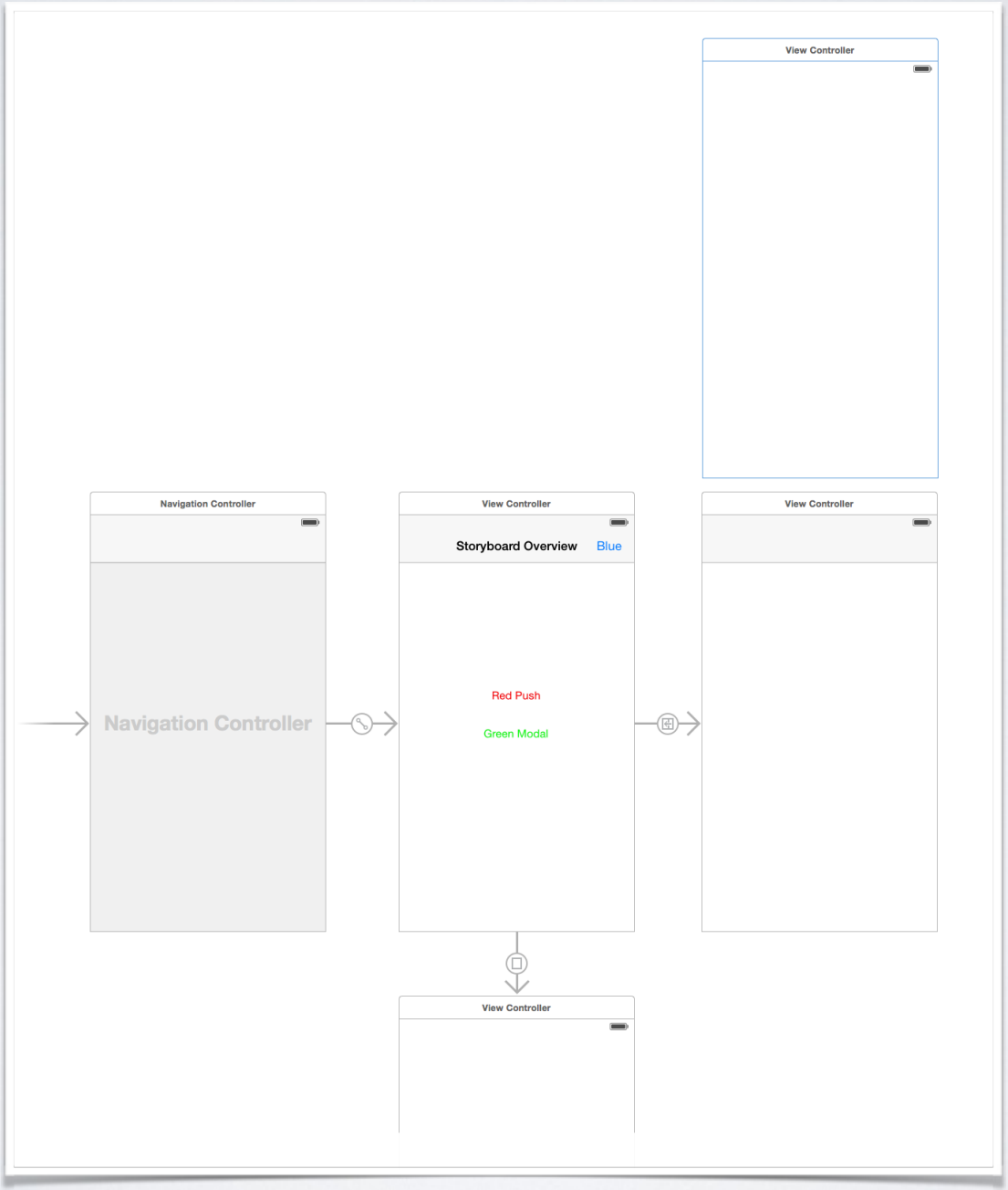
STEP 9: CHANGE THE TITLE OF THE NAVIGATION BAR



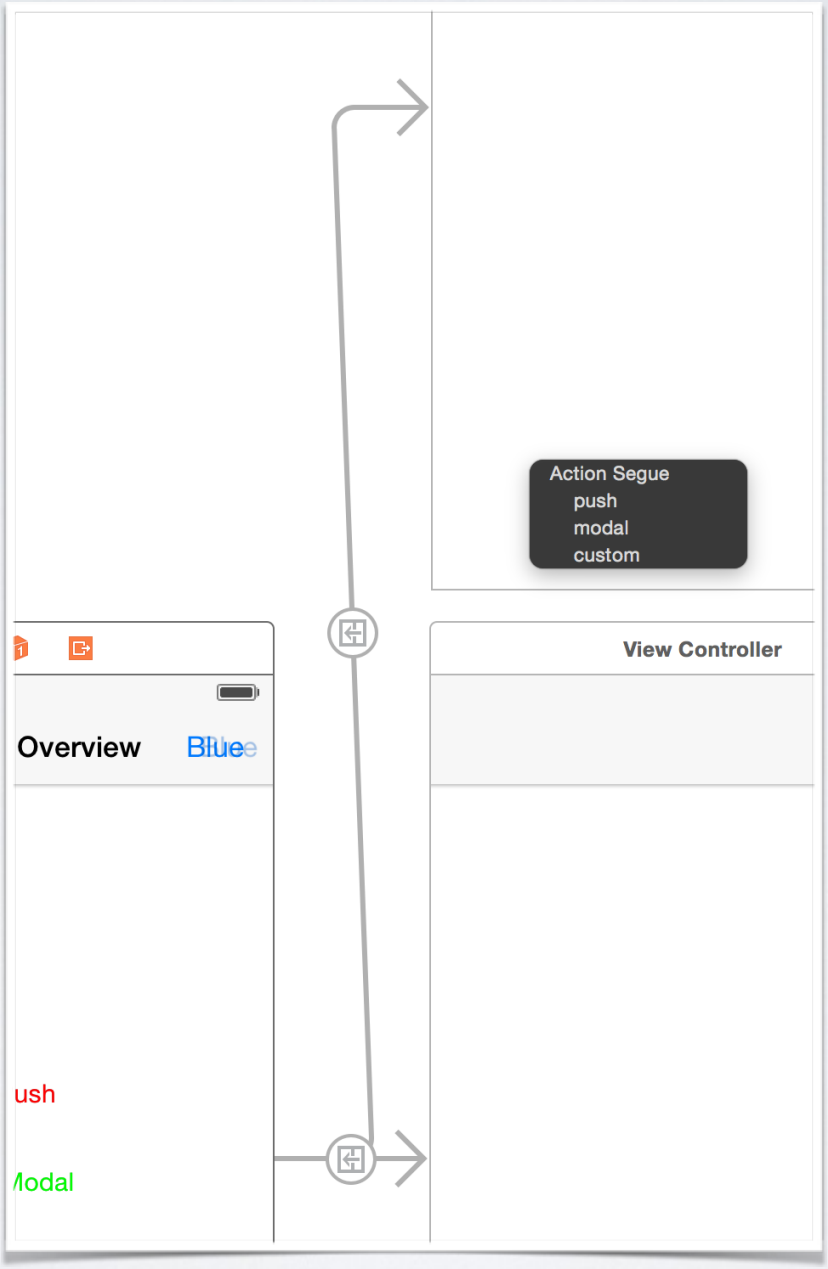
STEP 10: ADD A BAR BUTTON ITEM TO THE NAVIGATION BAR AND CHANGE THE TITLE TO BLUE



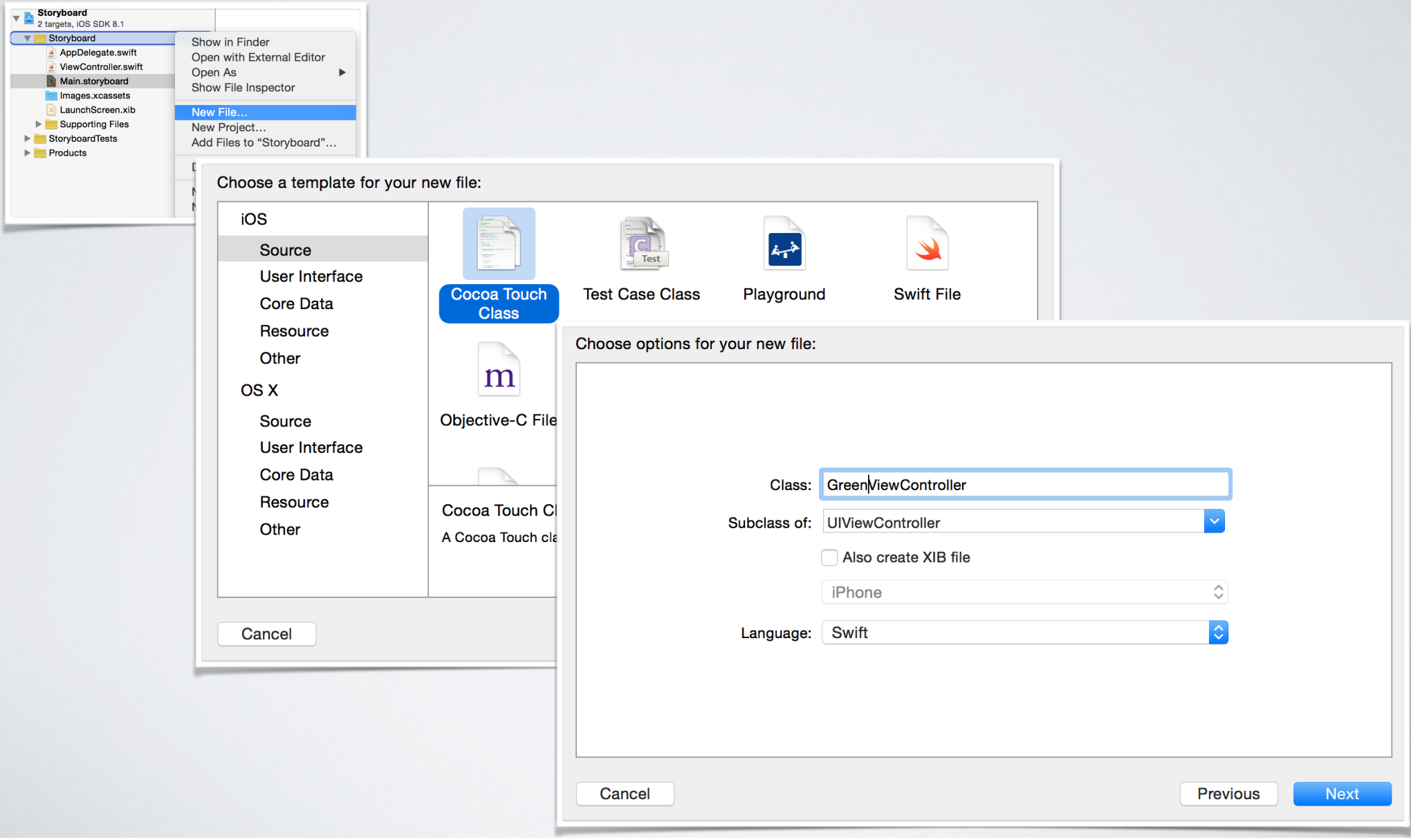
STEP 11: ADD OUR THIRD AND FINAL VIEW CONTROLLER TO THE STORYBOARD



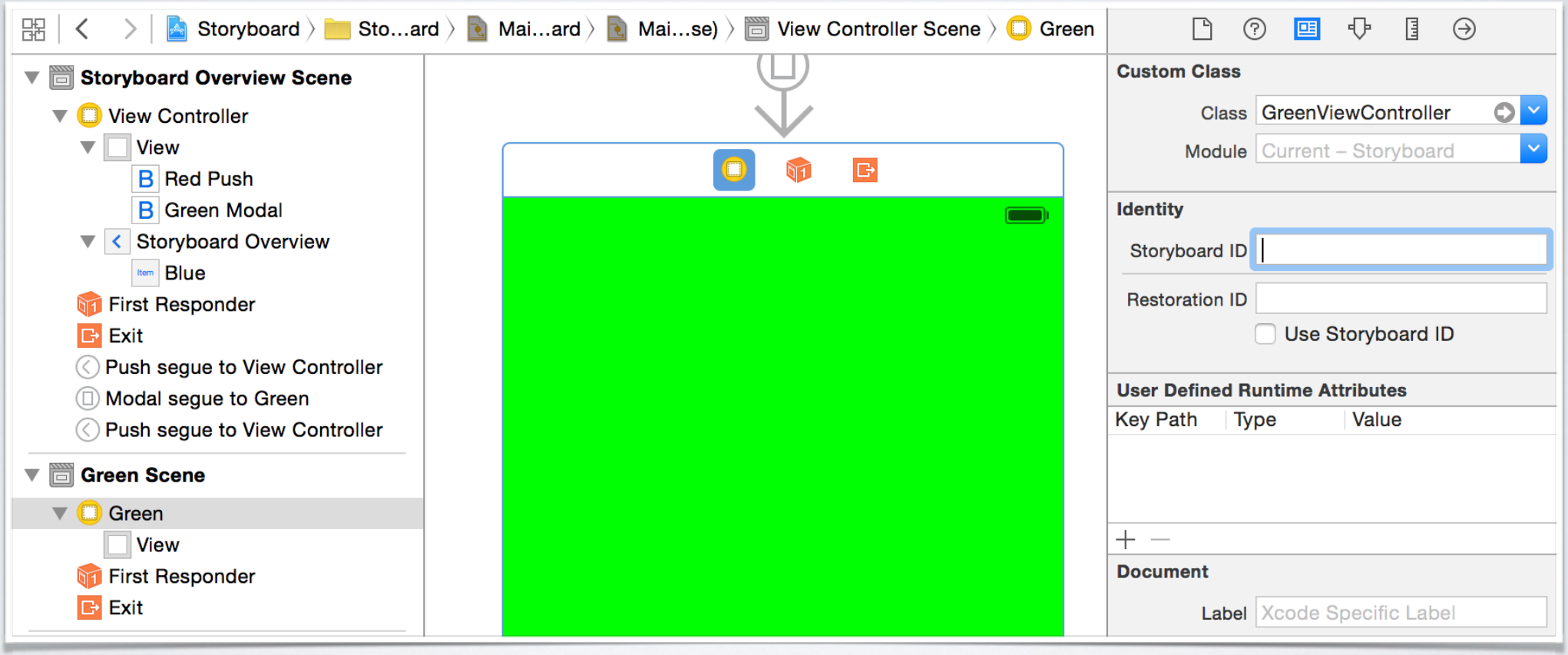
STEP 12: CONNECT OUR BAR BUTTON ITEM TO THE NEW VIEW CONTROLLER AND SELECT PUSH. REMEMBER TO HOLD DOWN THE [CONTROL] BUTTON WHILE CLICKING AND DRAGGING



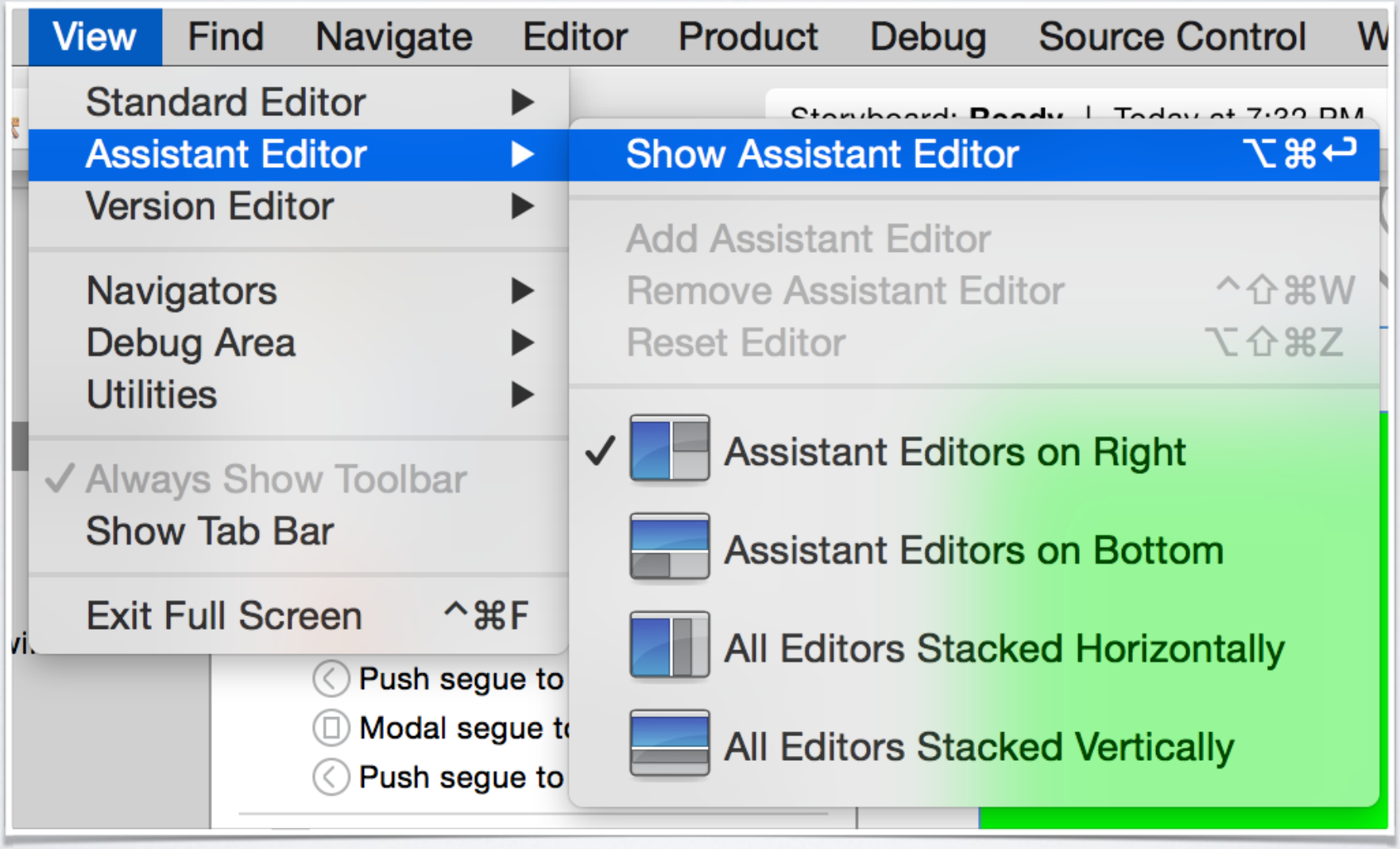
STEP 13: NOW WE NEED TO DO A LITTLE PROGRAMMING. ADD A CONTROLLER OBJECT AND NAME IT "GREENVIEWCONTROLLER" AS SHOWN BELOW



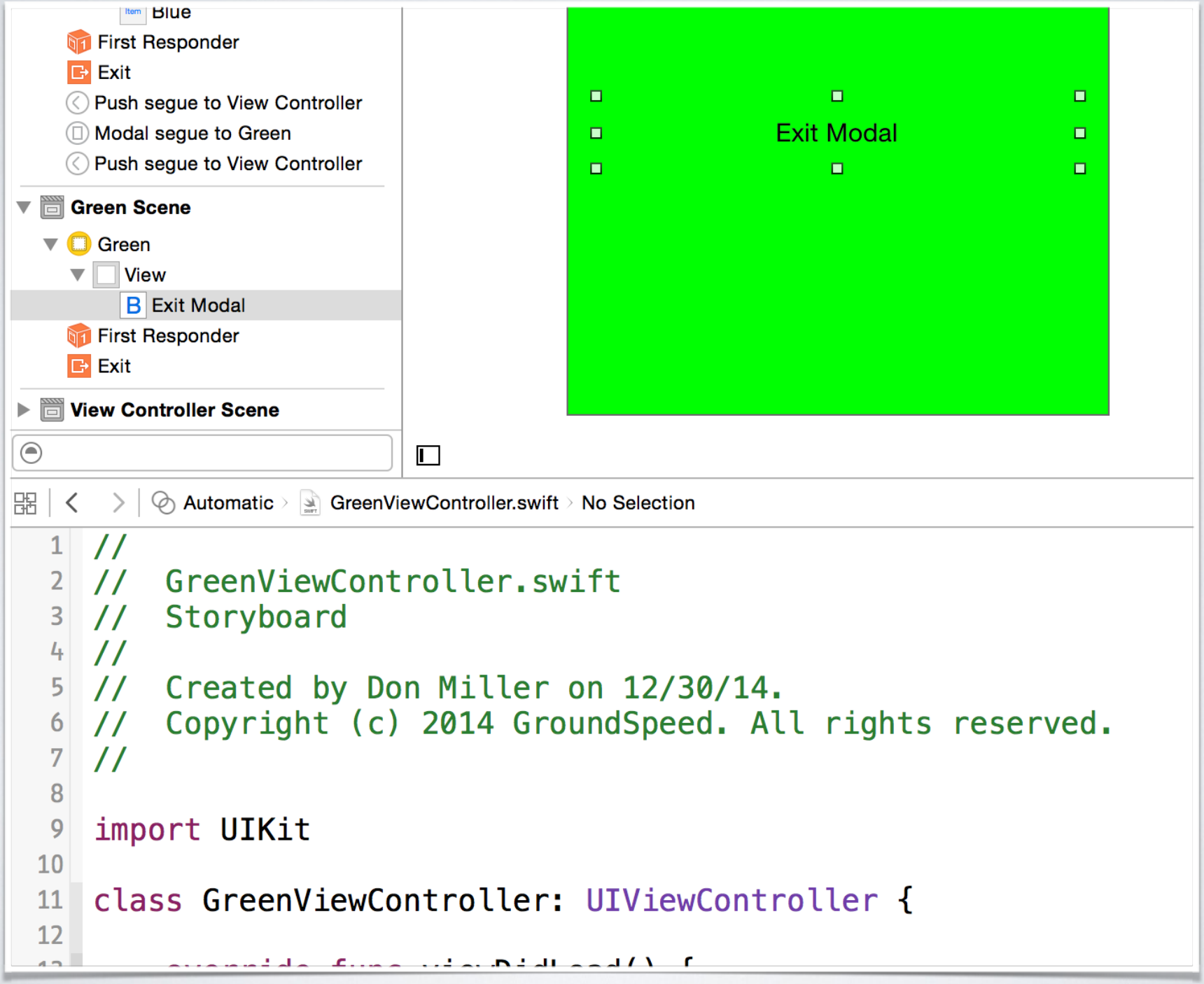
STEP 14: ASSIGN YOUR VIEW IN THE STORYBOARD TO THE NEW OBJECT VIEW CONTROLLER YOU JUST CREATED



STEP 15: SHOW THE ASSISTANT EDITOR TO HAVE OUR CODE GENERATED FOR US BY CLICKING

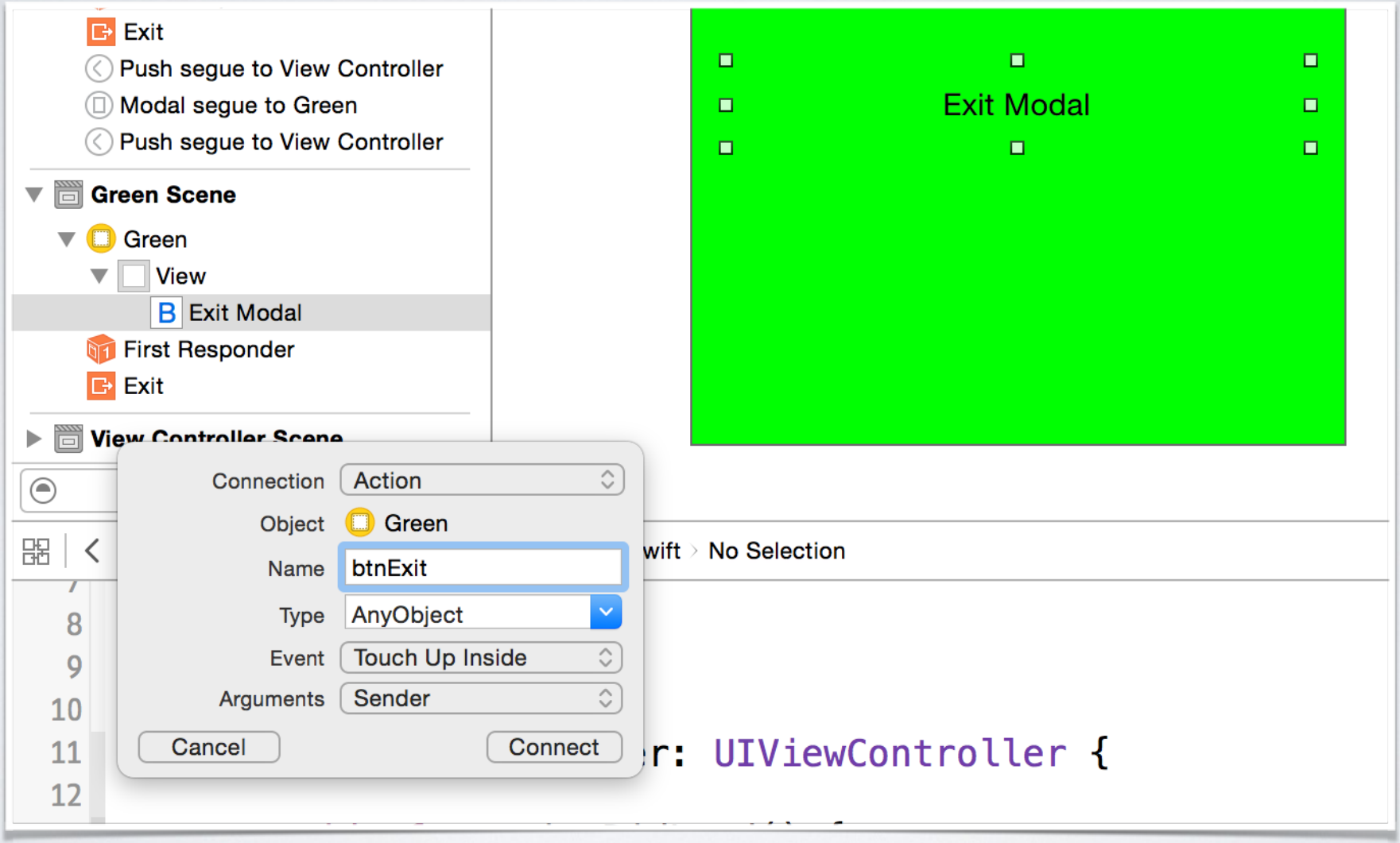


STEP 16: ADD A BUTTON TO THE GREEN VIEW



GroundSpeed™
rapid web + mobile software

STEP 17: HOLD DOWN THE CONTROL WHEN CLICKING FROM THE BUTTON TO YOU HEADER FILE AND DRAG IT UNDER THE INTERFACE LINE. CHANGE THE CONNECTION TO ACTION AND THE NAME TO BTNEXIT

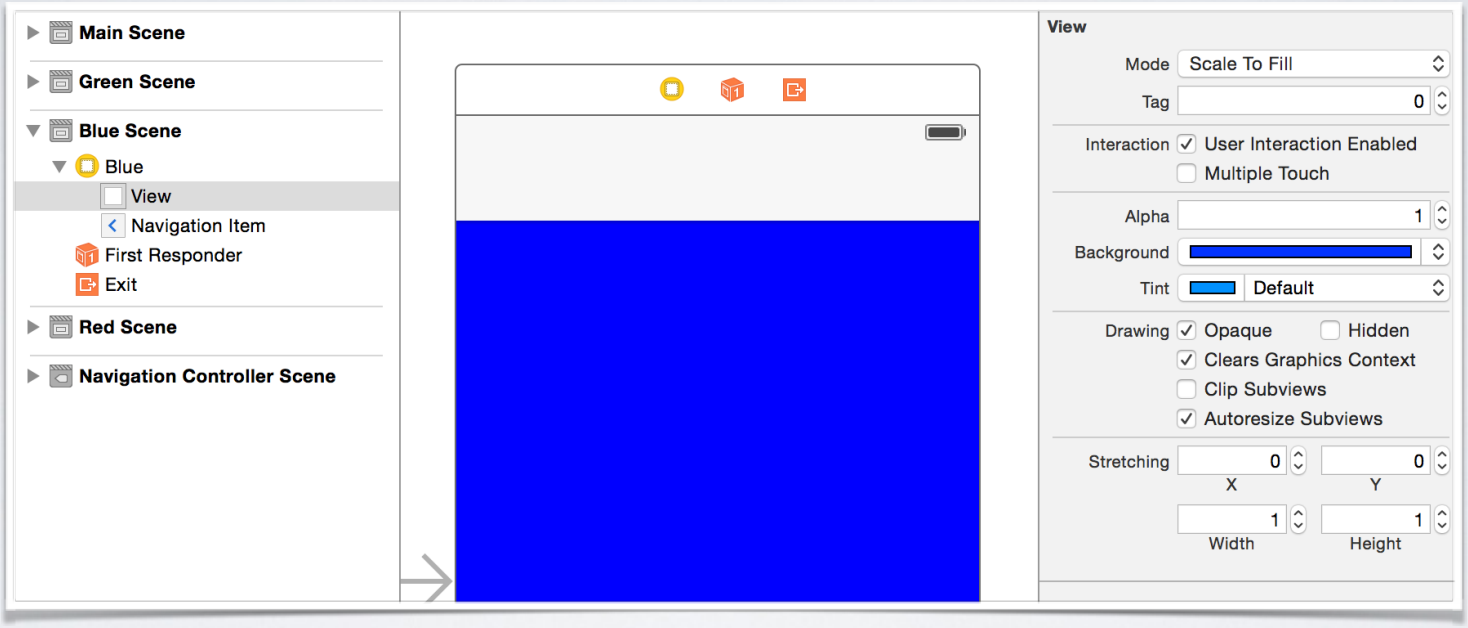


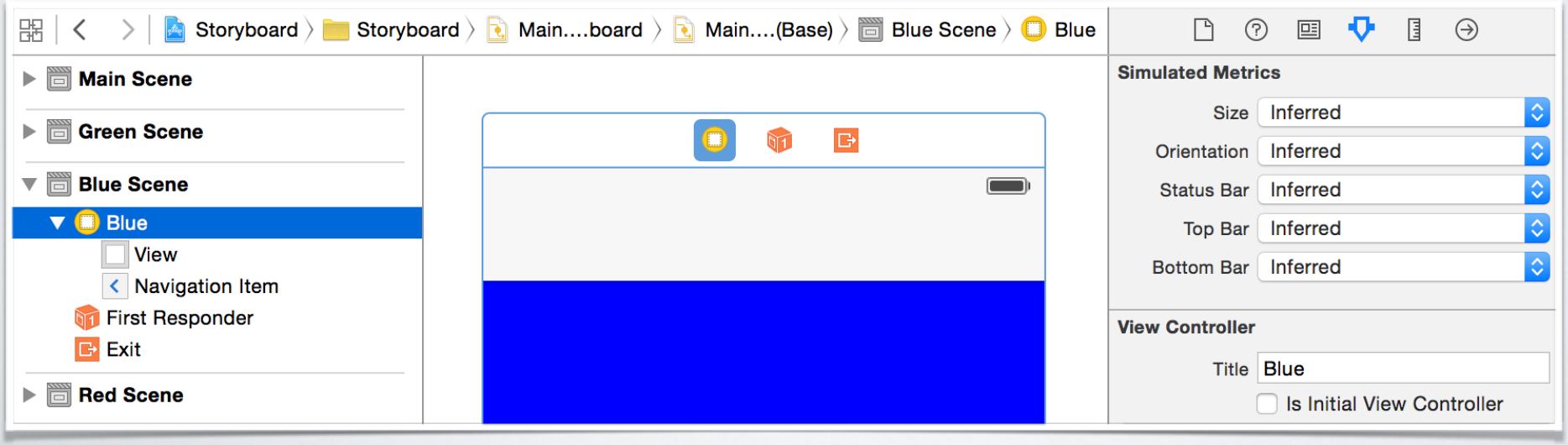
GroundSpeed™
rapid web + mobile software

STEP 18: ADD THE FOLLOWING LINE OF CODE IN THE BTNEXIT METHOD TO DISMISS THE MODAL WINDOW

```
 9  import UIKit
10
11  class GreenViewController: UIViewController {
12
13      override func viewDidLoad() {
14          super.viewDidLoad()
15
16          // Do any additional setup after loading the view.
17      }
18
19      override func didReceiveMemoryWarning() {
20          super.didReceiveMemoryWarning()
21          // Dispose of any resources that can be recreated.
22      }
23
24      @IBAction func btnExit(sender: AnyObject) {
25          self.dismissViewControllerAnimated(true, completion: nil)
26      }
27  }
```

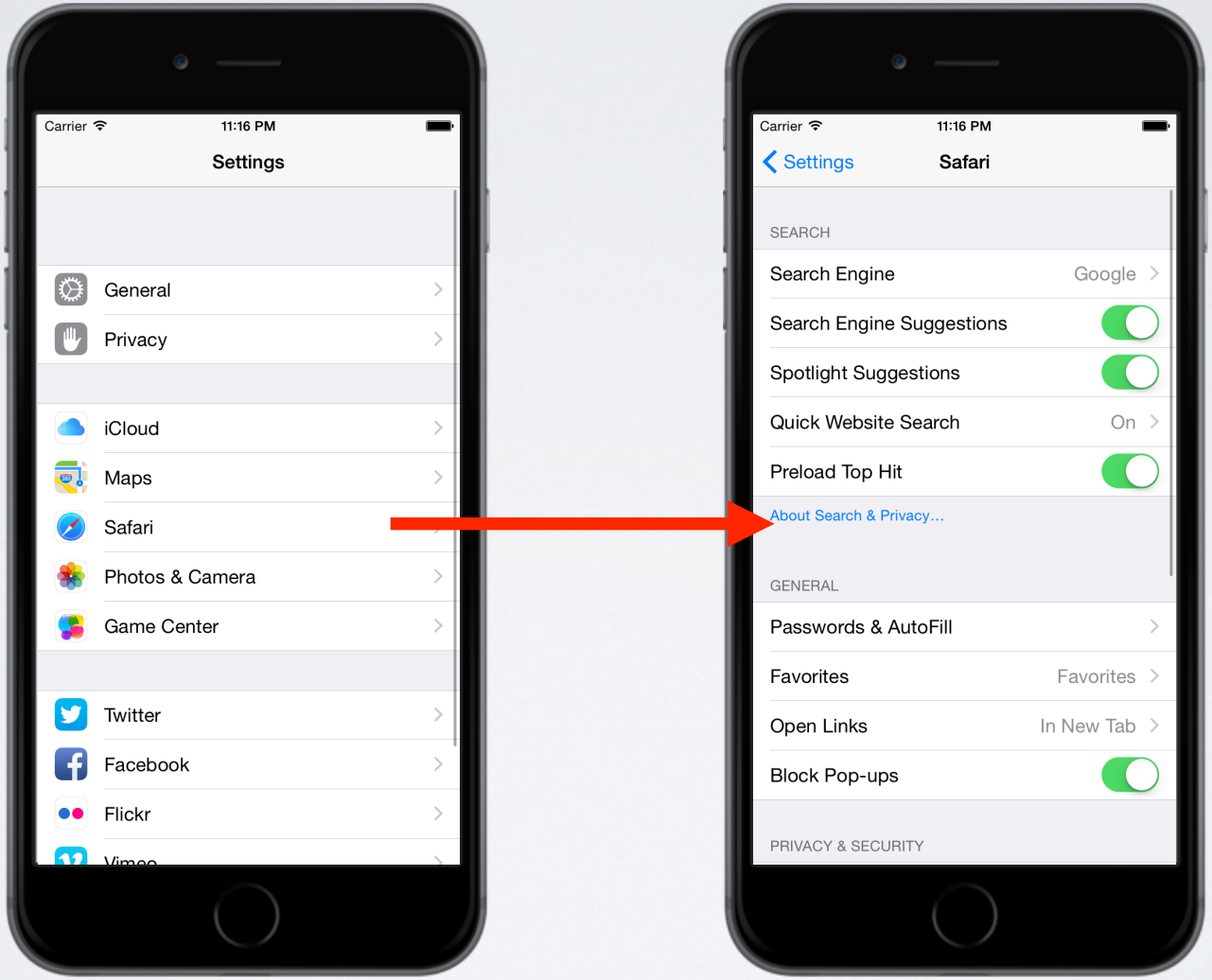
STEP 19: FINAL TOUCHES. CHANGE THE VIEW CONTROLLER TITLES AND BACKGROUNDS TO REFLECT THE COLOR DESCRIBED IN THE CALLING BUTTON





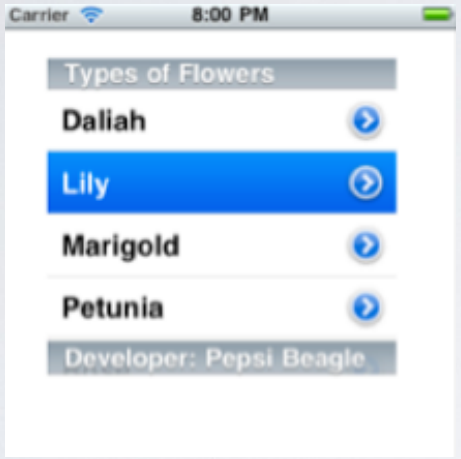
STEP 20: RUN IT!

UITABLEVIEW OVERVIEW

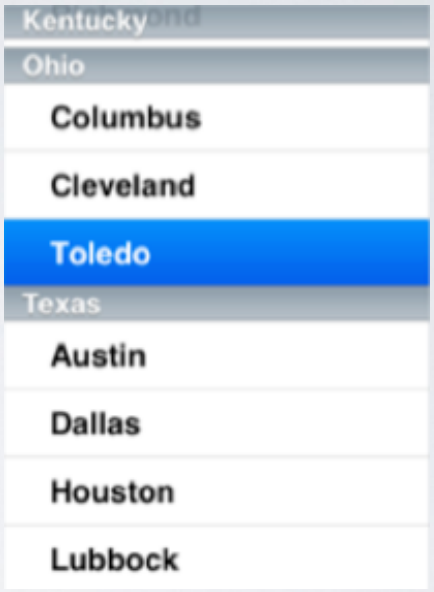


UITABLEVIEW OVERVIEW

A Simple UITableView with 4 Rows

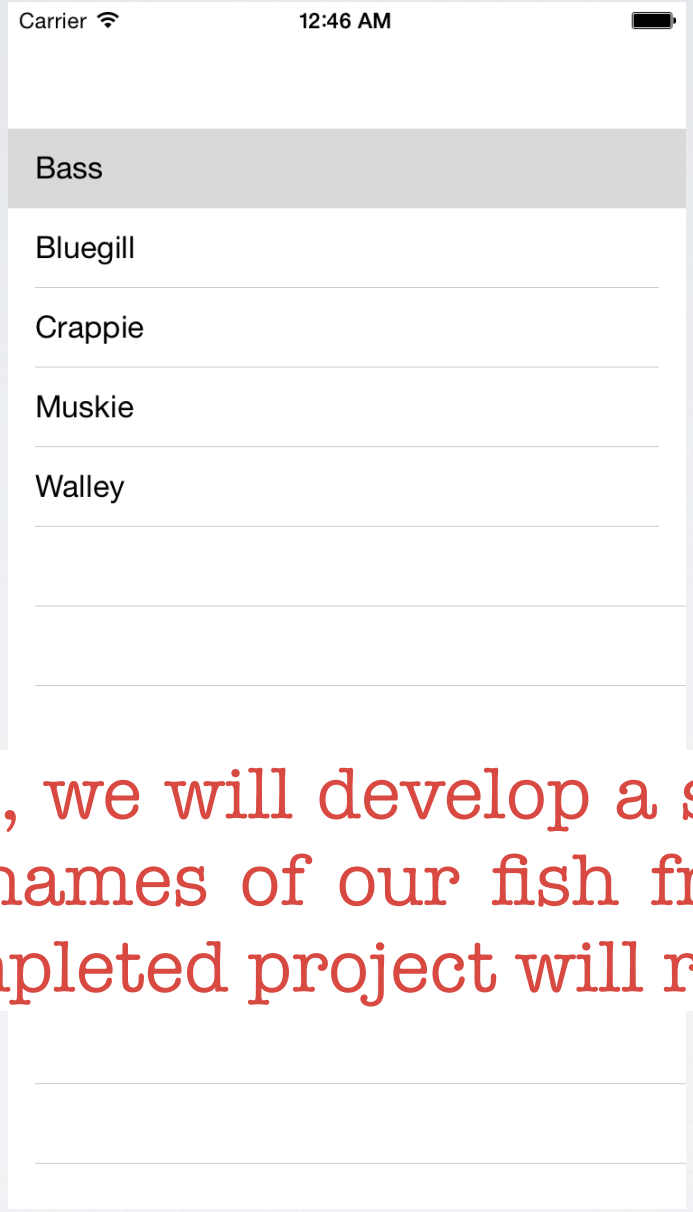


A Sectioned UITableView



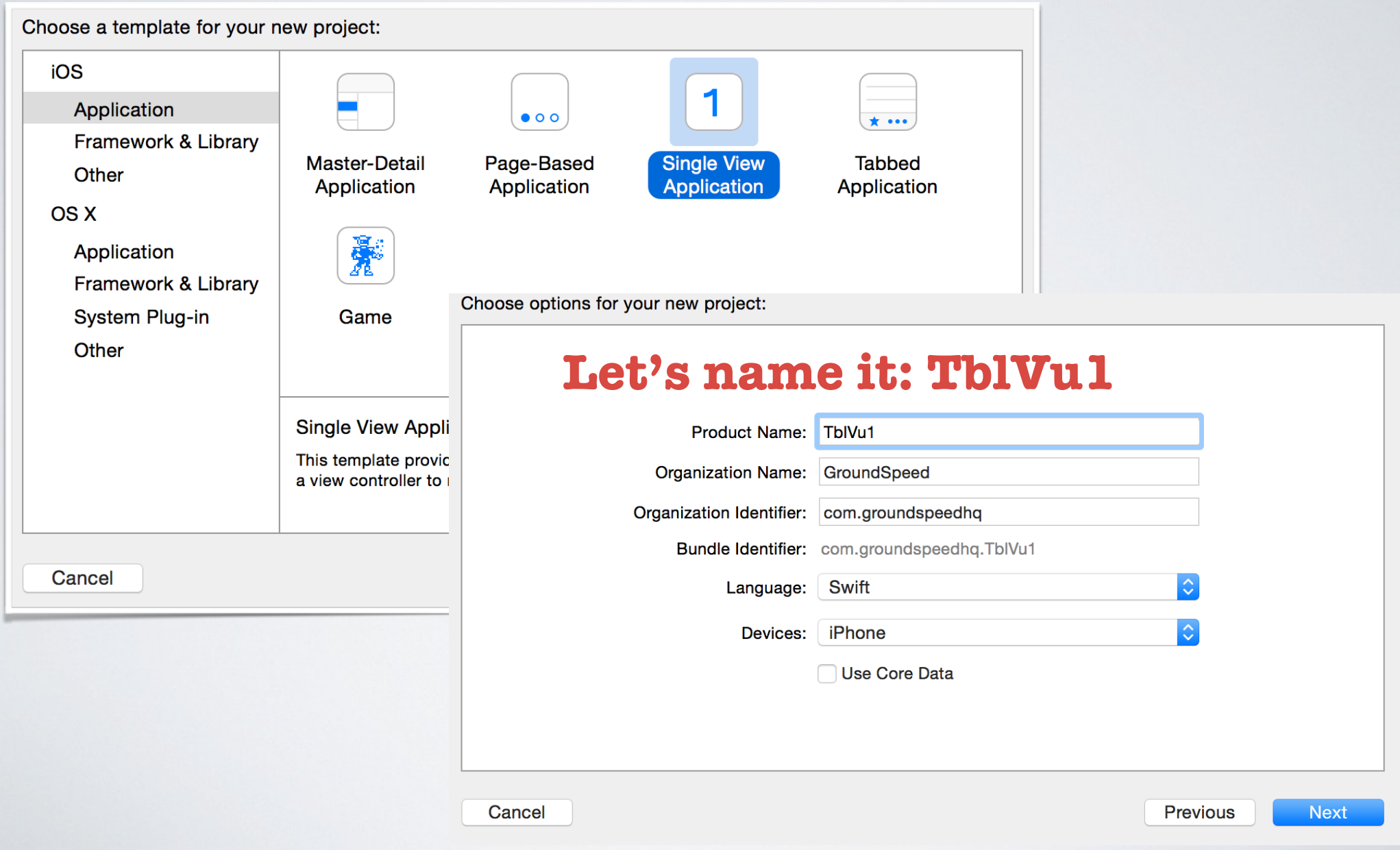
- Programmatically, the UIKit identifies rows and sections through their index number.
- Sections are numbered 0 through n-1 from the top of a table
- Rows are numbered 0 through n-1 within a section
- As Mentioned earlier, the table view comes in one of two basic styles: plain or grouped.

TABLE VIEW EXAMPLE

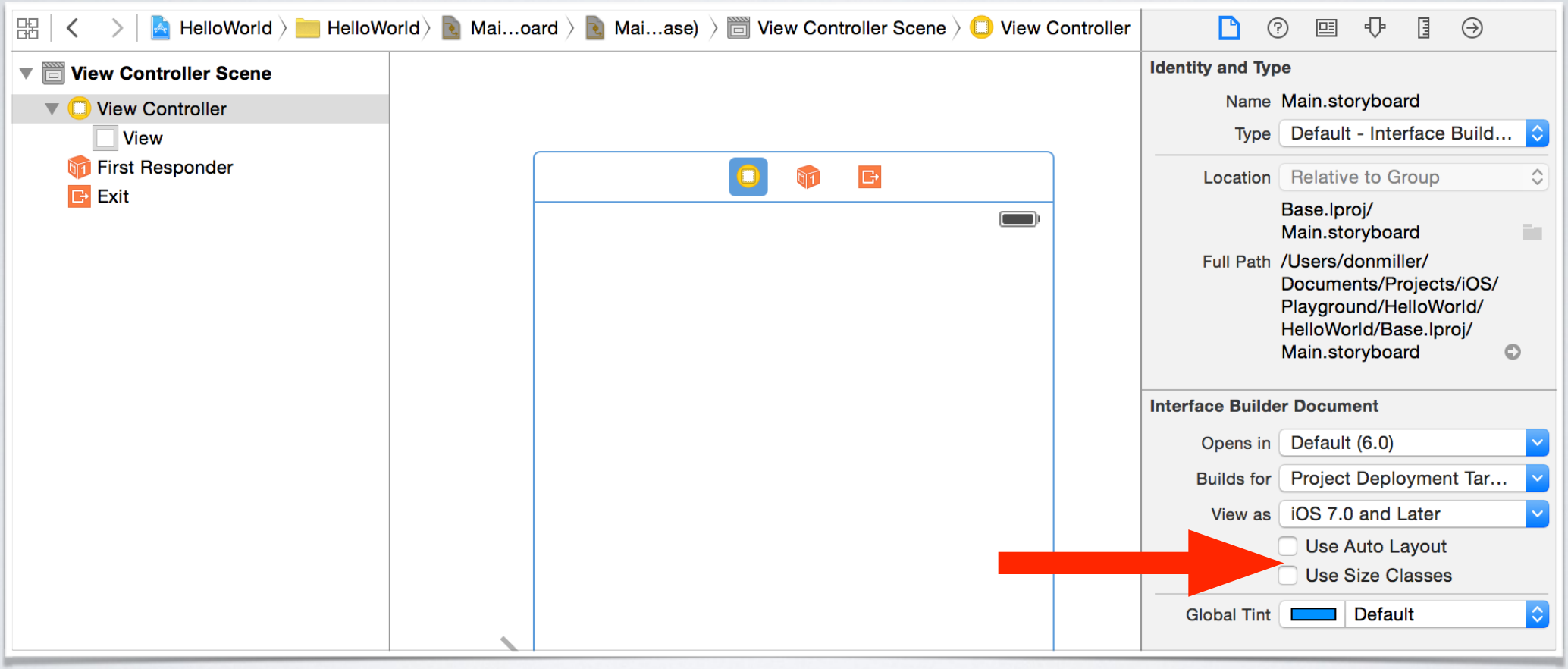


In this example, we will develop a simple table view to display the names of our fish from the example earlier. The completed project will run as shown.

STEP 1: CREATE A SINGLE VIEW BASED APP.

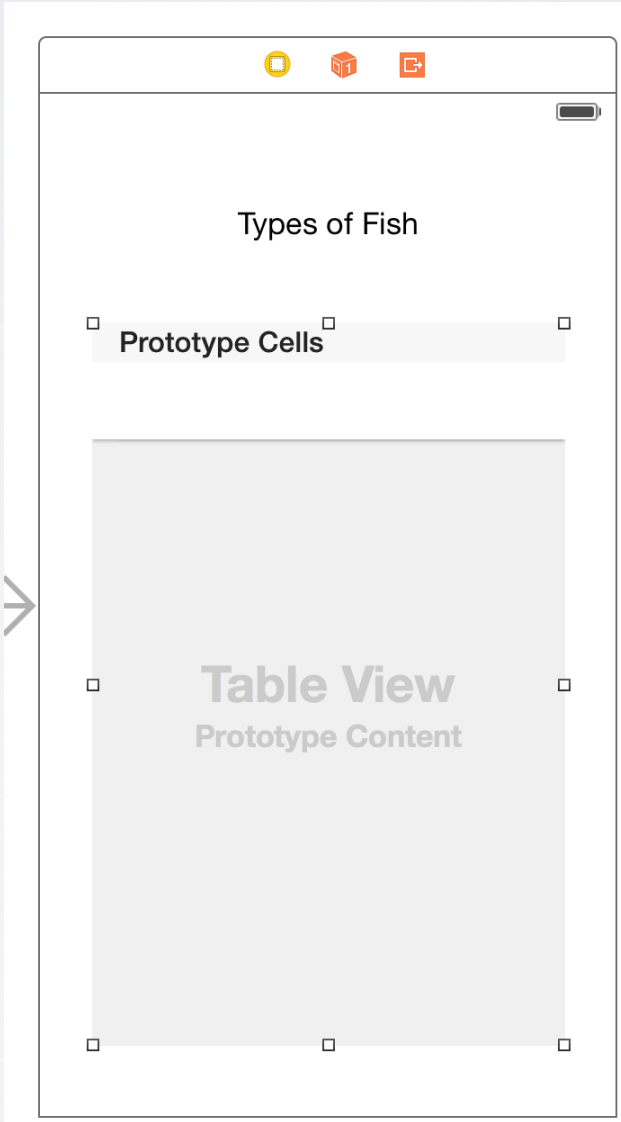


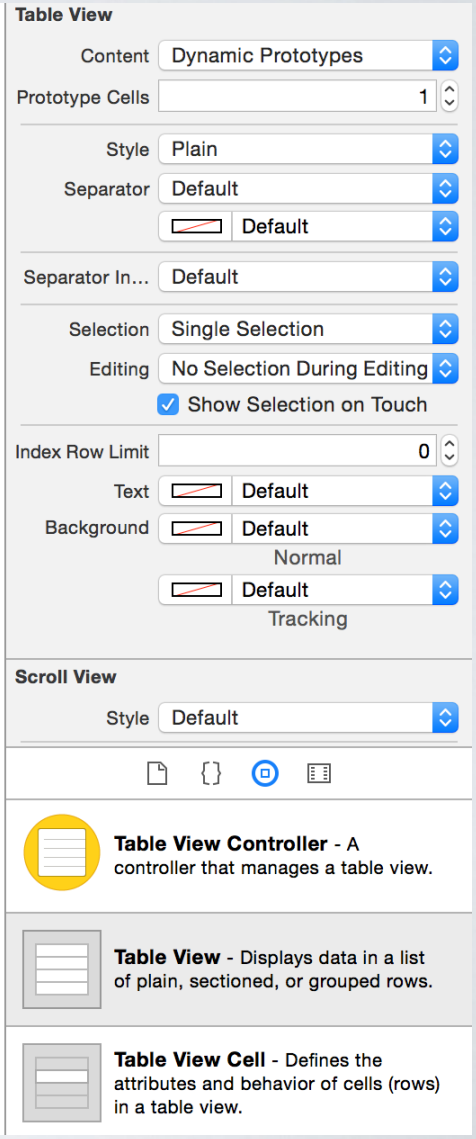
STEP 2: TURN OFF SIZE CLASSES



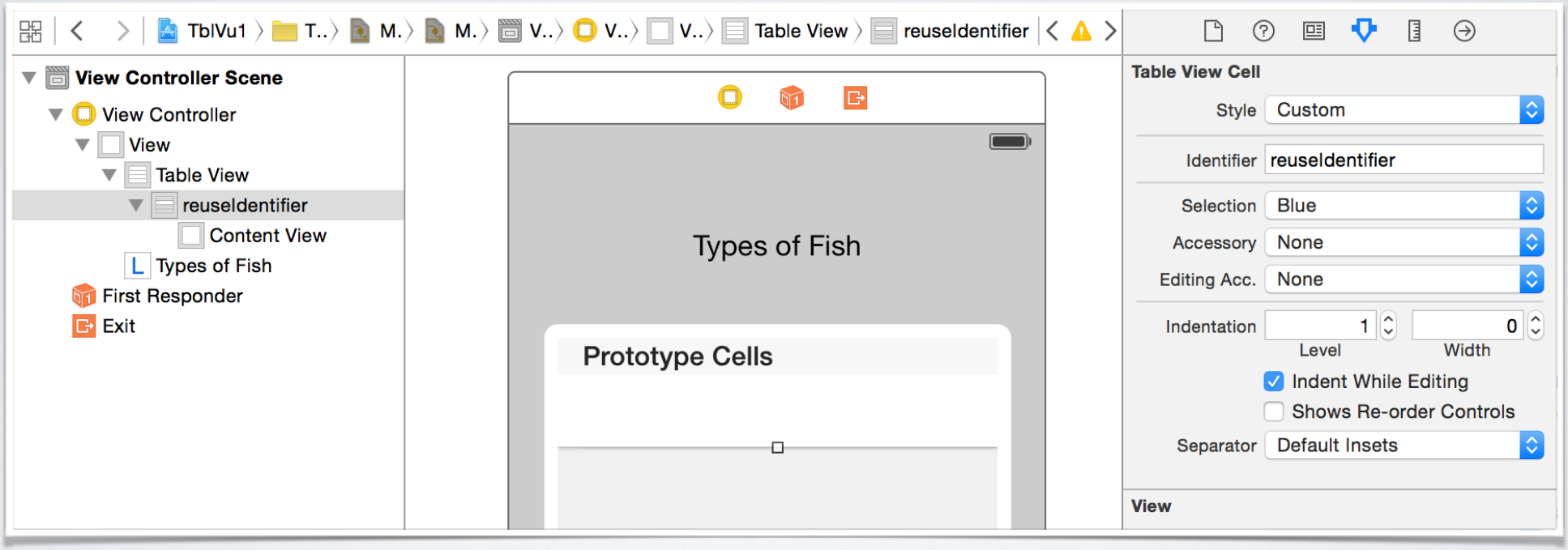
STEP 3: DRAW TABLE VIEW

- **Open the Main.storyboard file.**
- **Draw a label and change its text to "Types of Fish".**
- **Now draw a TableView as shown**



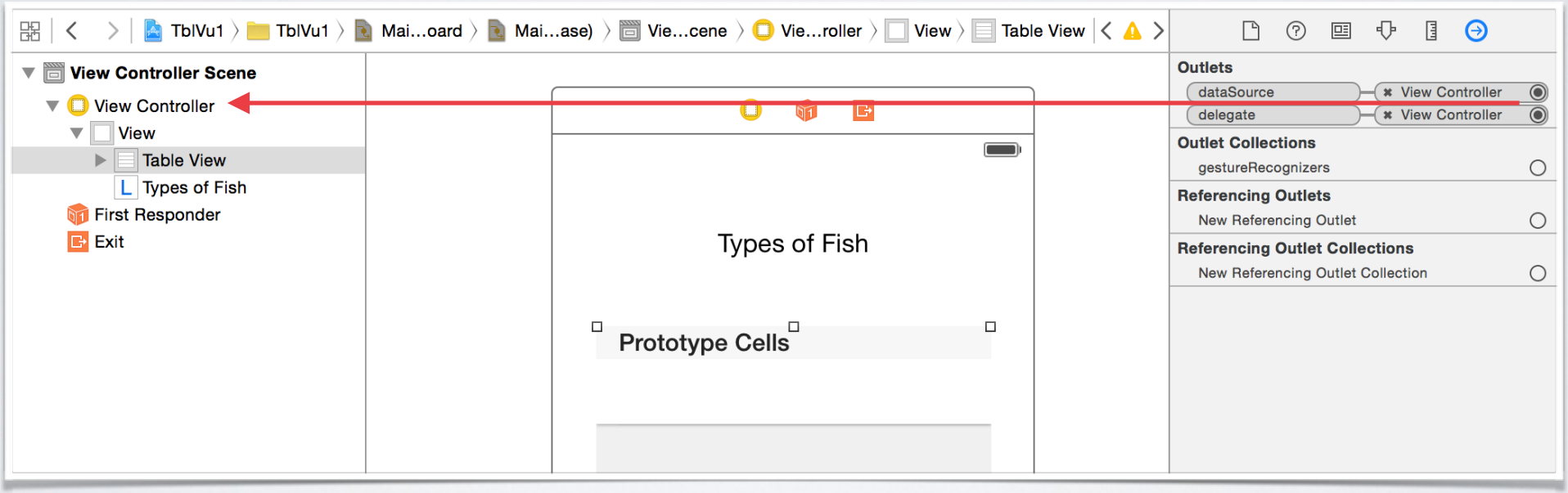


STEP 4: UPDATE PROTOTYPE CELL WITH REUSE IDENTIFIER



STEP 5: DATASOURCE & DELEGATE

Open the Property Inspector of the Table View, then drag its dataSource and delegate outlets and drop those on the File's Owner.



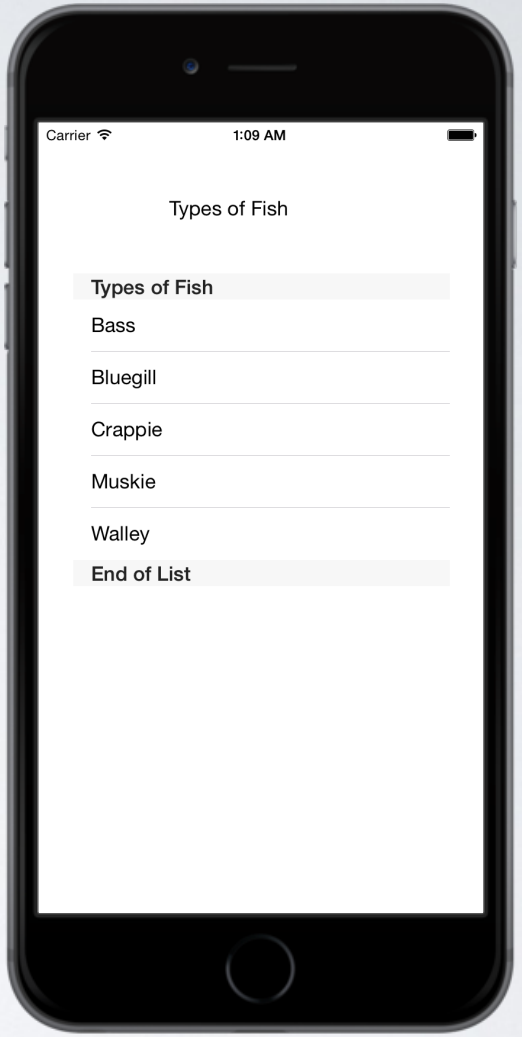
STEP 6: OPEN THE VIEWCONTROLLER.SWIFT FILE AND INSERT THE FOLLOWING CODE

```
class ViewController: UIViewController, UITableViewDelegate, UITableViewDataSource {

    let arrayFish = ["Bass", "Bluegill", "Crappie", "Muskie", "Walley"]

    override func viewDidLoad() {
        super.viewDidLoad()
    }

    func numberOfSectionsInTableView(tableView: UITableView) -> Int {
        return 1
    }

    func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return self.arrayFish.count
    }

    func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath) ->
UITableViewCell {
        let cell = tableView.dequeueReusableCellWithIdentifier("reuseIdentifier", forIndexPath:
indexPath) as UITableViewCell

        // Configure the cell...
        cell.textLabel?.text = self.arrayFish[indexPath.row]

        return cell
    }
}
```

STEP 7: OPTIONAL HEADER AND FOOTER TITLE OF SECTIONS: YOU MAY IMPLEMENT THE FOLLOWING METHODS FOR THE HEADER AND FOOTER TITLES

```
func tableView(tableView: UITableView, titleForHeaderInSection section: Int) -> String?
{
    return "Types of Fish"
}

func tableView(tableView: UITableView, titleForFooterInSection section: Int) -> String?
{
    return "End of List"
}
```

YOU ARE DONE!



GroundSpeed™
rapid web + mobile software

Don Miller

don@GroundSpeedHQ.com

@donmiller

http://github.com/donmiller

http://github.com/groundspeed

GroundSpeed™
rapid web + mobile software