1

Build Your First App

"The way to get started is to quit talking and begin doing."

– *Walt Disney*

Hello World! Build Your First App Using Swift

By now you should have installed Xcode 6 and some understandings of Swift language. If you haven't done so, check out the previous chapter about what you need to begin iOS programming. We'll use Xcode 6.3 (or up) to work on all exercises in this book.

You may have heard of the "Hello World" program if you have read any programming book before. Hello World is a program for the first-time programmer to create. It's a very simple program that outputs "Hello, World" on the screen of a device.

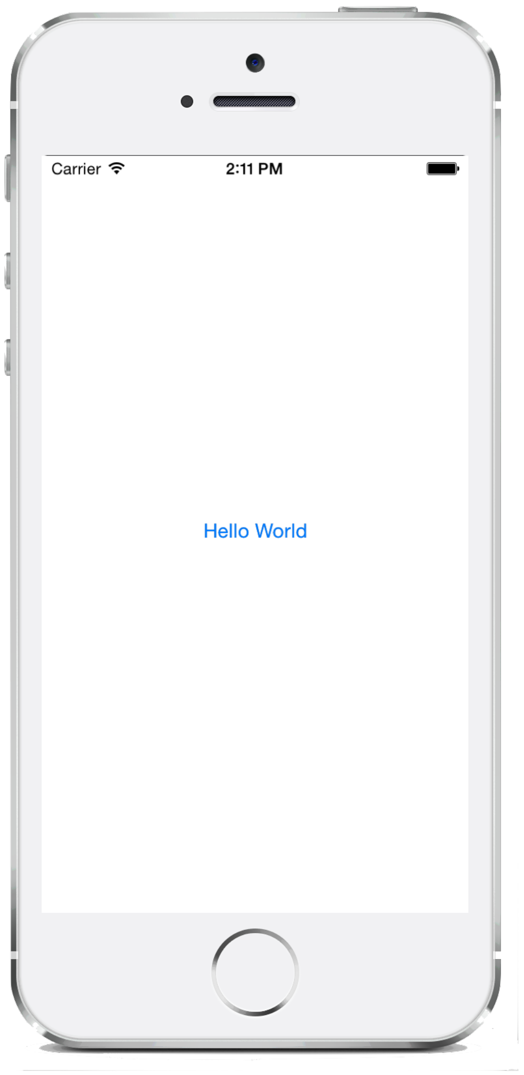


Figure 1-1. HelloWorld App

It's a tradition in the programming world. So, let's follow the programming tradition and create a "Hello World" app using Xcode. Despite its simplicity, the "Hello World" program serves a few purposes:

- It gives you an overview about the syntax and structure of Swift, the new programming language of iOS.
- It also gives you a basic introduction to the Xcode 6 environment. You'll learn how to create an Xcode project and lay out your user interface using Storyboard. Even if you've used Xcode 5 before, you'll learn what's new in the latest version of Xcode.
- You'll learn how to compile a program, build the app and test it using the Simulator.
- Lastly, it makes you think programming is not difficult. I don't want to scare you away from learning programming. It'll be fun.

Your First App

Your first app, as displayed in figure 1-1, is very simple and just shows a "Hello World" button. When user taps the button, the app shows a welcome message. That's it. Extremely simple but it helps you kick off your iOS programming journey.

Let's Jump Right Into Create a Project

First, launch Xcode. If you've installed Xcode via the Mac App Store, you should be able to locate Xcode in the LaunchPad. Just click on the Xcode icon to start it up.

Once launched, Xcode displays a welcome dialog. From here, choose "Create a new Xcode project" to start a new project:

It's normal if you do not understand the source code. Just relax and focus on building your first app. Familiarize yourself with the Xcode environment and Storyboard. I will explain the language as we go along and you will learn how the HelloWorld app works in the next chapter.

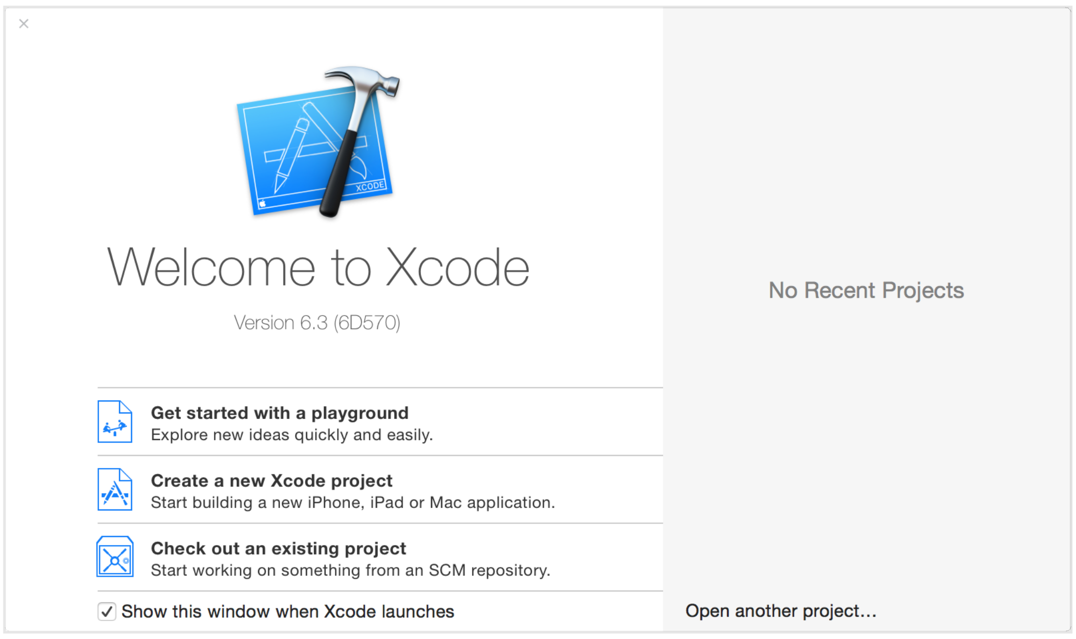
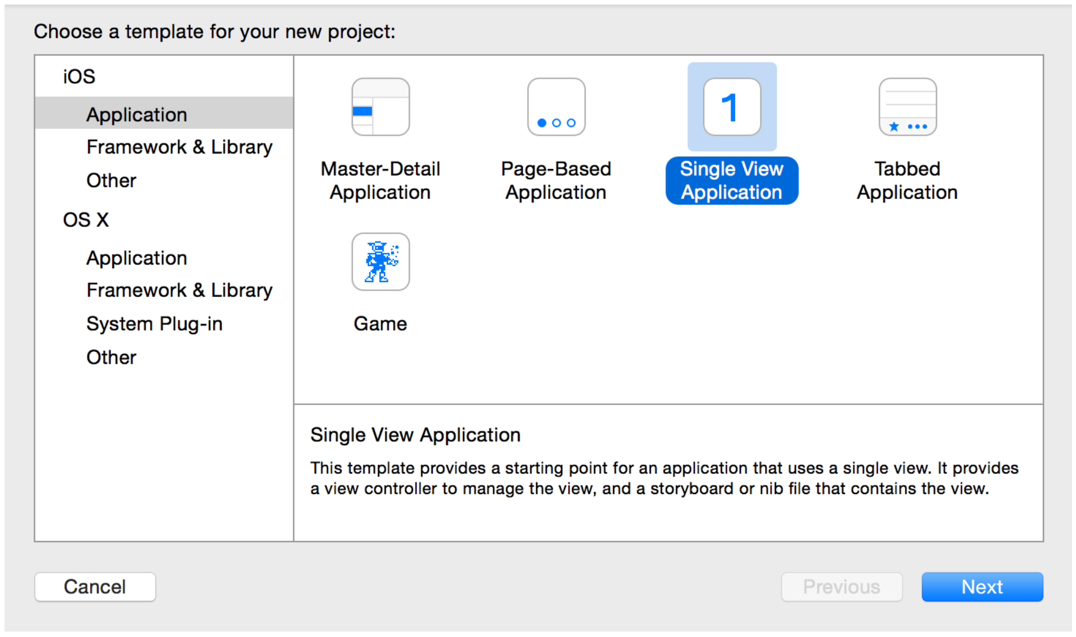

Welcome to Xcode
Version 6.3 (6D570)
Get started with a playground
Explore new ideas quickly and easily.
Create a new Xcode project
Start building a new iPhone, iPad or Mac application.
Check out an existing project
Start working on something from an SCM repository.
✓ Show this window when Xcode launches
No Recent Projects
Open another project…

Figure 1-2. Xcode - Welcome Dialog

Xcode shows various project templates for selection. For your first app, choose "Single View Application" and click "Next".

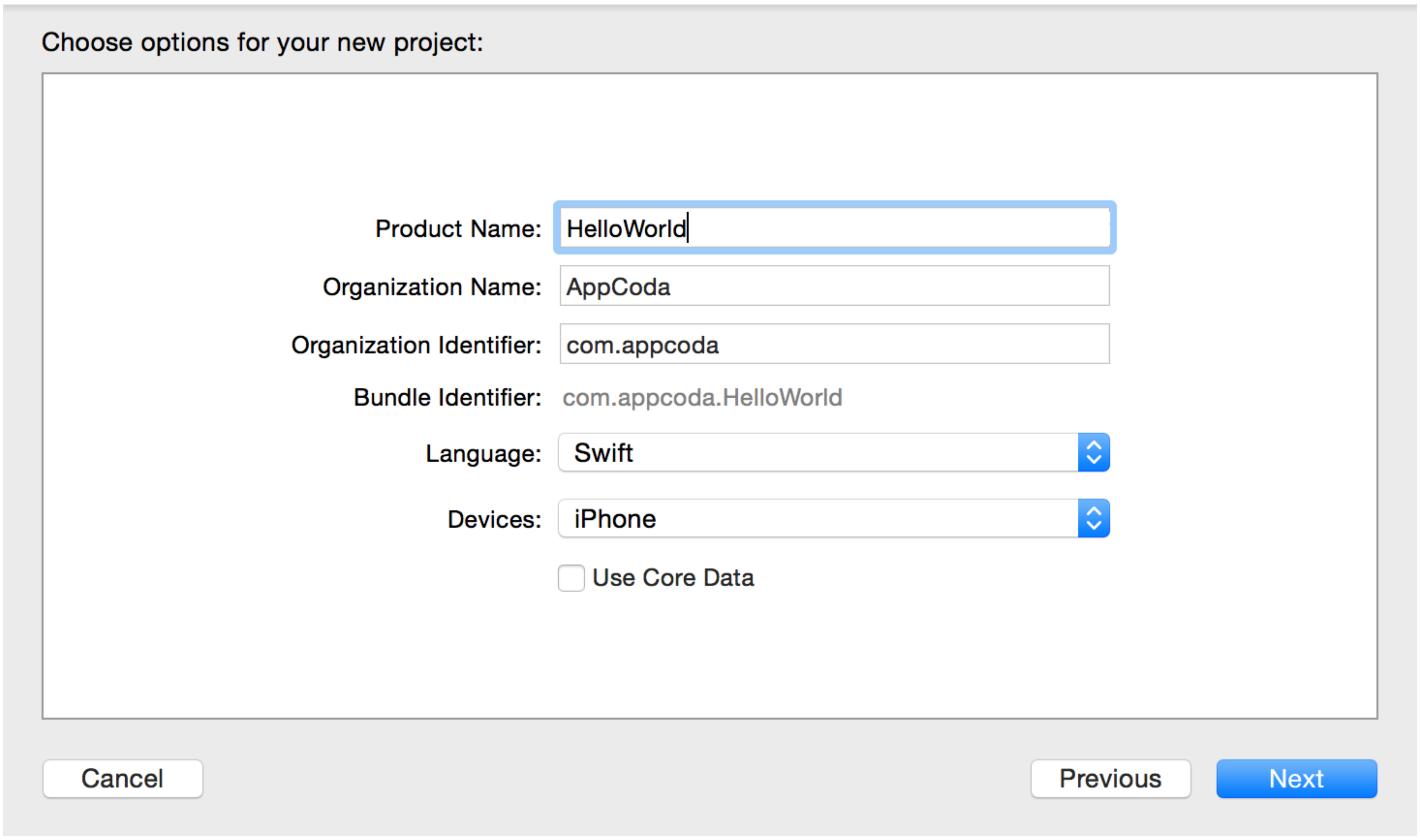


Single View Application

This template provides a starting point for an application that uses a single view. It provides a view controller to manage the view, and a storyboard or nib file that contains the view.

Figure 1-3. Xcode Project Template Selection

This brings you to the next screen to fill in all the necessary options for your project.



Product Name: HelloWorld
Organization Name: AppCoda
Organization Identifier: com.appcoda
Bundle Identifier: com.appcoda.HelloWorld
Language: Swift
Devices: iPhone
Use Core Data [ ]

Figure 1-4. Options for your Hello World project

You can simply fill in the options as follows:

Product Name: *HelloWorld* – This is the name of your app.

Organization Name: *AppCoda* – It's the name of your organization.

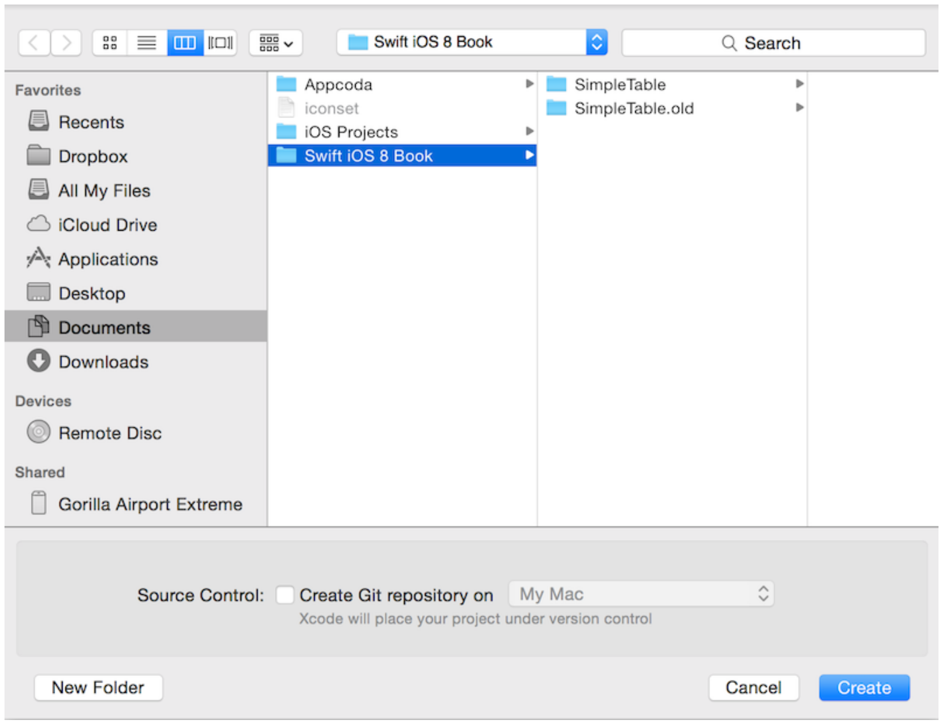
Organization Identifier: *com.appcoda* – It's actually the domain name written the other way round. If you have a domain, you can use your own domain name. Otherwise, you may use "com.appcoda" or just fill in "edu.self".

Bundle Identifier: *com.appcoda.HelloWorld* - It's a unique identifier of your app, which is used during app submission. You do not need to fill in this option. Xcode automatically generates it for you.

Language: *Swift* – Xcode 6 supports both Objective-C and Swift for app development. As this book is about Swift, we'll use Swift to develop the project.

Devices: *iPhone* – Select “iPhone” for this project.

Use Core Data: *[unchecked]* – Do not select this option. You do not need Core Data for this simple project. We’ll explain Core Data in later chapters.

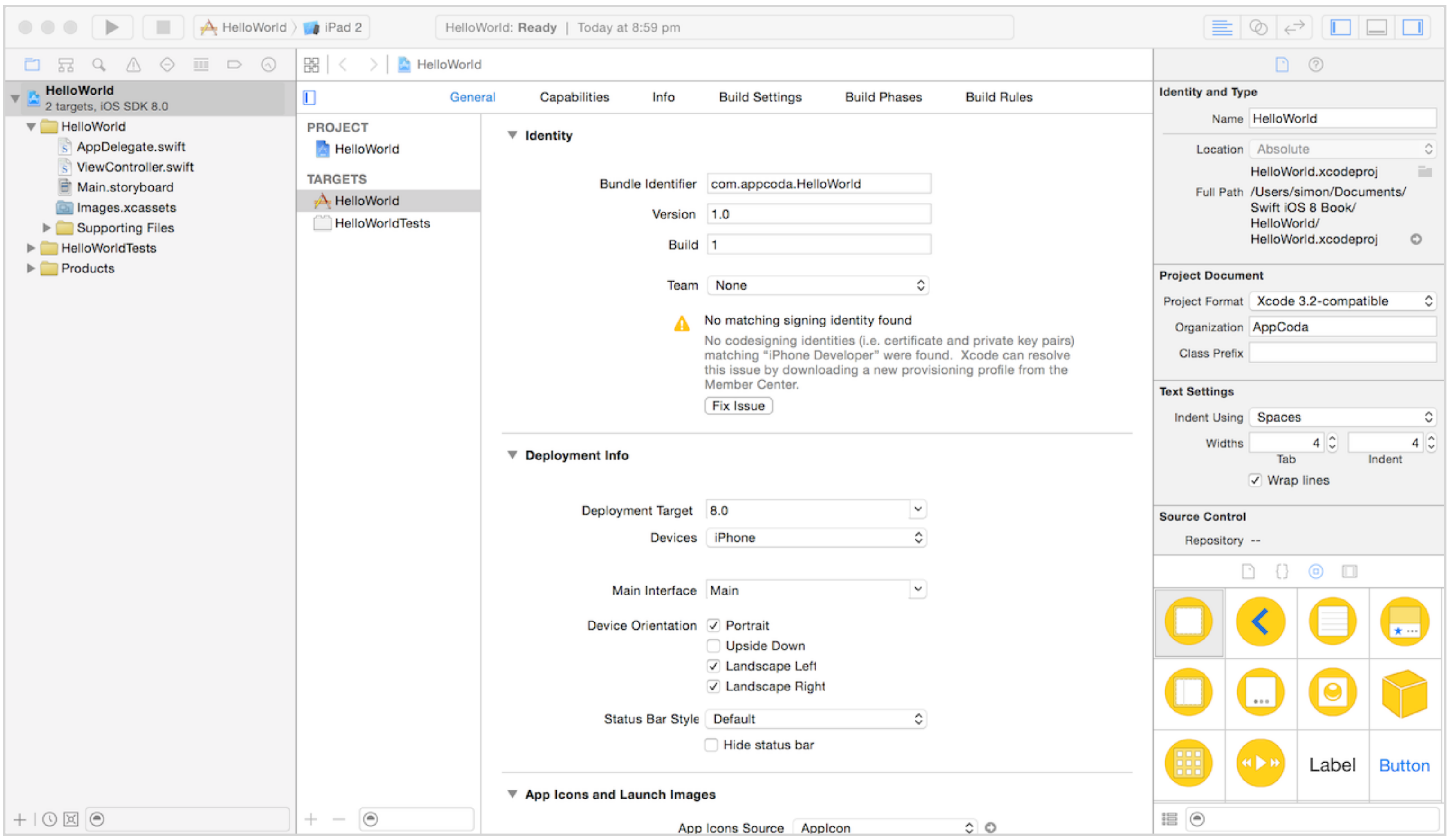


Source Control: ☐ Create Git repository on My Mac
Xcode will place your project under version control

Figure 1-5. Choose a folder and save your project

Click “Next” to continue. Xcode then asks you where to save the “HelloWorld” project. Pick any folder (e.g. Desktop) on your Mac. You may notice there is an option for source control. Just deselect it. We do not need to use the option in this book. Click “Create” to continue.

After you confirm, Xcode automatically creates the “Hello World” project. The screen will look like the screenshot shown in figure 1-6. You can ignore the “No matching signing identity found” error.



Bundle Identifier: com.appcoda.HelloWorld
Version: 1.0
Build: 1
Team: None

No matching signing identity found
No codesigning identities (i.e. certificate and private key pairs) matching “iPhone Developer” were found. Xcode can resolve this issue by downloading a new provisioning profile from the Member Center.

Deployment Target: 8.0
Devices: iPhone
Main Interface: Main
Device Orientation: ✓ Portrait, ☐ Upside Down, ✓ Landscape Left, ✓ Landscape Right
Status Bar Style: Default
☐ Hide status bar
App Icons Source: AppIcon

Identity and Type
Name: HelloWorld
Location: Absolute
HelloWorld.xcodeproj
Full Path: /Users/simon/Documents/Swift iOS 8 Book/HelloWorld/HelloWorld.xcodeproj

Project Document
Project Format: Xcode 3.2-compatible
Organization: AppCoda
Class Prefix:

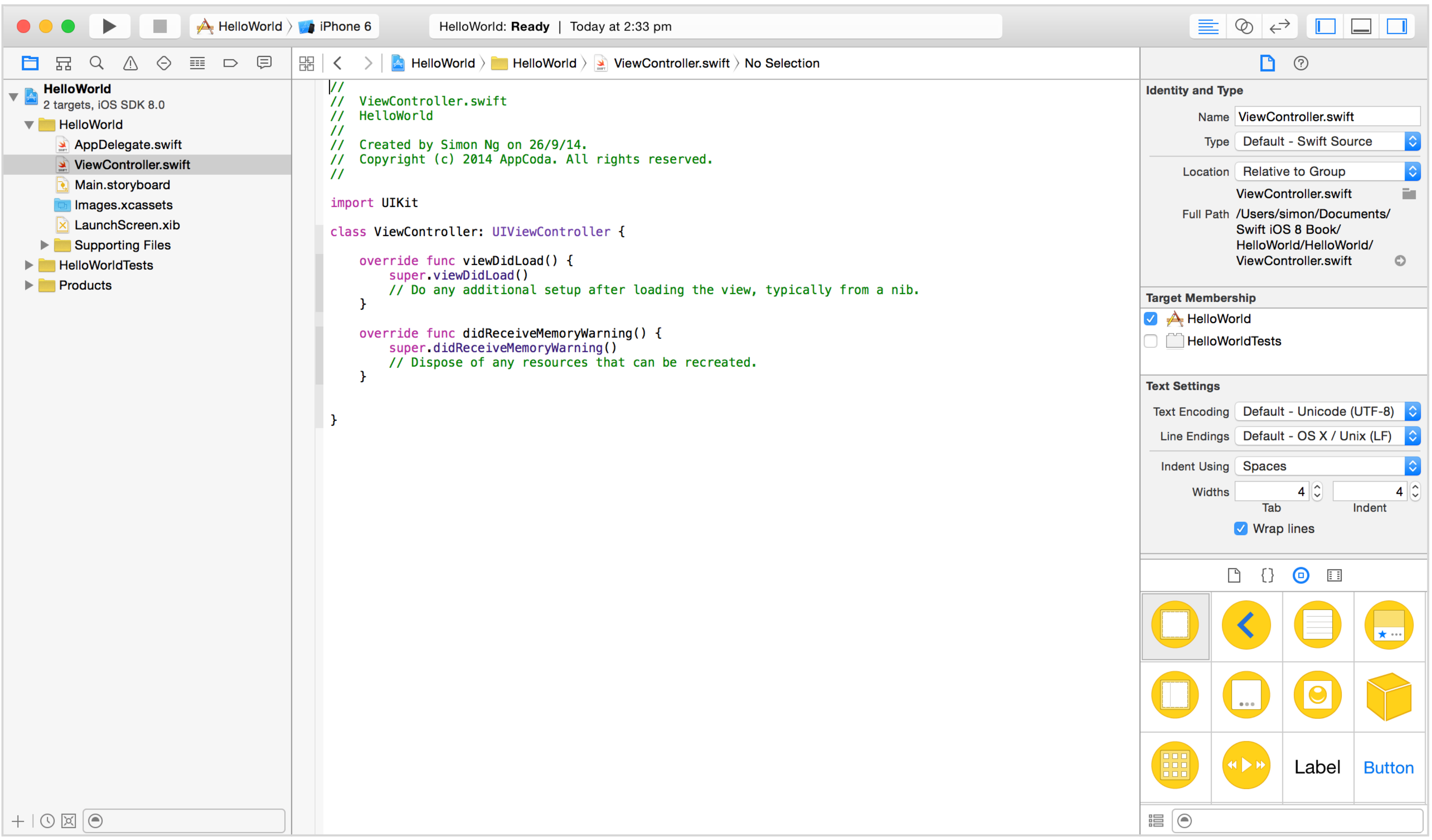
Text Settings
Indent Using: Spaces
Widths: 4 Tab, 4 Indent
✓ Wrap lines

Source Control
Repository --

Figure 1-6. Main Xcode Window for HelloWorld Project

Familiarize Yourself with Xcode Workspace

Before we move on to the coding part, let’s take a few minutes to have a quick look at the Xcode workspace environment. In the left pane is the project navigator. You can find all your project files in this area. The center part of the workspace is the editor area. You do all the editing stuff here (such as editing the project setting, source code file, user interface) in this area. Depending on the type of file, Xcode shows you different interfaces in the editor area. For instance, if you select ViewController.swift in the project navigator, Xcode displays the source code in the center area (see figure 1-7). If you select the “Main.storyboard”, which is the file for storing user interface, Xcode shows you the visual editor for storyboard (see figure 1-8).



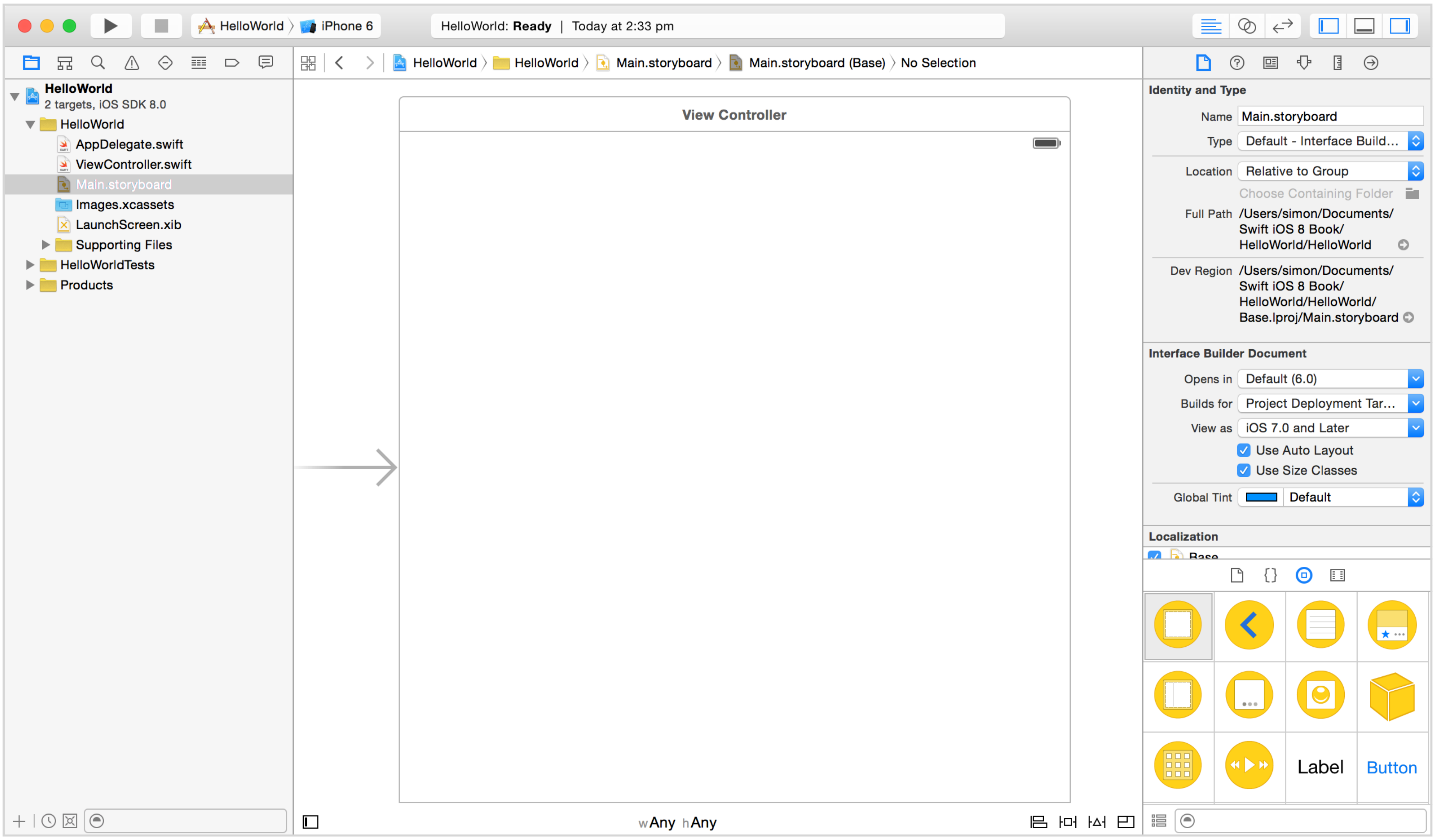
```
//
//  ViewController.swift
//  HelloWorld
//
//  Created by Simon Ng on 26/9/14.
//  Copyright (c) 2014 AppCoda. All rights reserved.
//

import UIKit

class ViewController: UIViewController {

    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.
    }


}
```

Identity and Type
Name: ViewController.swift
Type: Default - Swift Source
Location: Relative to Group
ViewController.swift
Full Path: /Users/simon/Documents/Swift iOS 8 Book/HelloWorld/HelloWorld/ViewController.swift

Target Membership
☑ HelloWorld
☐ HelloWorldTests

Text Settings
Text Encoding: Default - Unicode (UTF-8)
Line Endings: Default - OS X / Unix (LF)
Indent Using: Spaces
Widths: 4 (Tab) 4 (Indent)
☑ Wrap lines

Figure 1-7. Xcode Workspace with Source Code Editor


HelloWorld 〉 iPhone 6 | HelloWorld: **Ready** | Today at 2:33 pm

HelloWorld 〉 HelloWorld 〉 Main.storyboard 〉 Main.storyboard (Base) 〉 No Selection

- HelloWorld
  2 targets, iOS SDK 8.0
  - HelloWorld
    - AppDelegate.swift
    - ViewController.swift
    - Main.storyboard
    - Images.xcassets
    - LaunchScreen.xib
    - Supporting Files
  - HelloWorldTests
  - Products

View Controller

wAny hAny

Identity and Type

Name	Main.storyboard
Type	Default - Interface Build...
Location	Relative to Group
	Choose Containing Folder
Full Path	/Users/simon/Documents/Swift iOS 8 Book/HelloWorld/HelloWorld
Dev Region	/Users/simon/Documents/Swift iOS 8 Book/HelloWorld/HelloWorld/Base.lproj/Main.storyboard

Interface Builder Document

Opens in	Default (6.0)
Builds for	Project Deployment Tar...
View as	iOS 7.0 and Later
	✓ Use Auto Layout
	✓ Use Size Classes
Global Tint	Default

Localization

✓ Base

Label Button

Figure 1-8 . Xcode Workspace with Storyboard Editor

The rightmost pane is the utility area. This area displays the properties of the file and allows you to access Quick Help. If Xcode doesn't show this area, you can select the rightmost button in the toolbar to enable it.

The middle view button of the view selector is deselected by default. If you click on it, Xcode displays the debug area right below the editor area. The debug area, as its name suggests, is used for showing debug messages. We'll talk about that in a later chapter, so don't worry if you do not understand what each area is for.

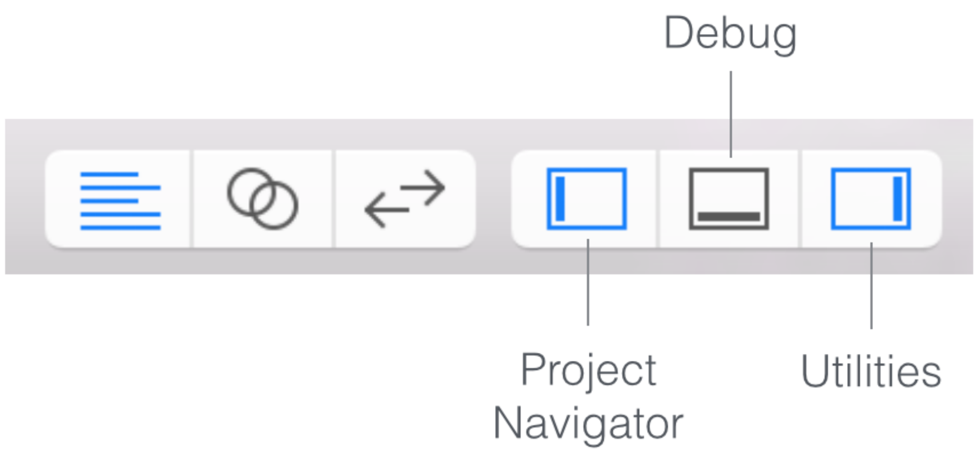


Figure 1-9. Show/hide the content areas of your workspace

Run Your App for the First Time

Until now, we have written zero lines of code. Even so, you can run your app using the built-in Simulator. This will give you an idea how to build and test your app in Xcode. In the toolbar you should see the Run button. If you hit the Run button, Xcode automatically builds the app and runs it in the selected Simulator. By default, the Simulator is set to iPhone 6. If you click the iPhone 6 button, you'll see a list of available Simulators. As we're going to build an iPhone app, you can use iPhone 6 (or any other iPhone models) as the Simulator. For this demo, I chose the iPhone 5s. Once selected, you can click the Run button to load your app in the Simulator. Figure 1-11 shows the simulator for an iPhone 5s.

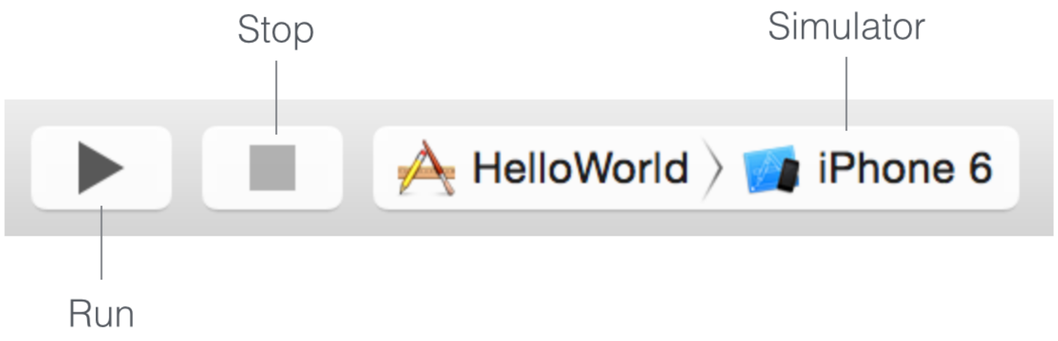


Figure 1-10. Run and Stop Buttons in Xcode

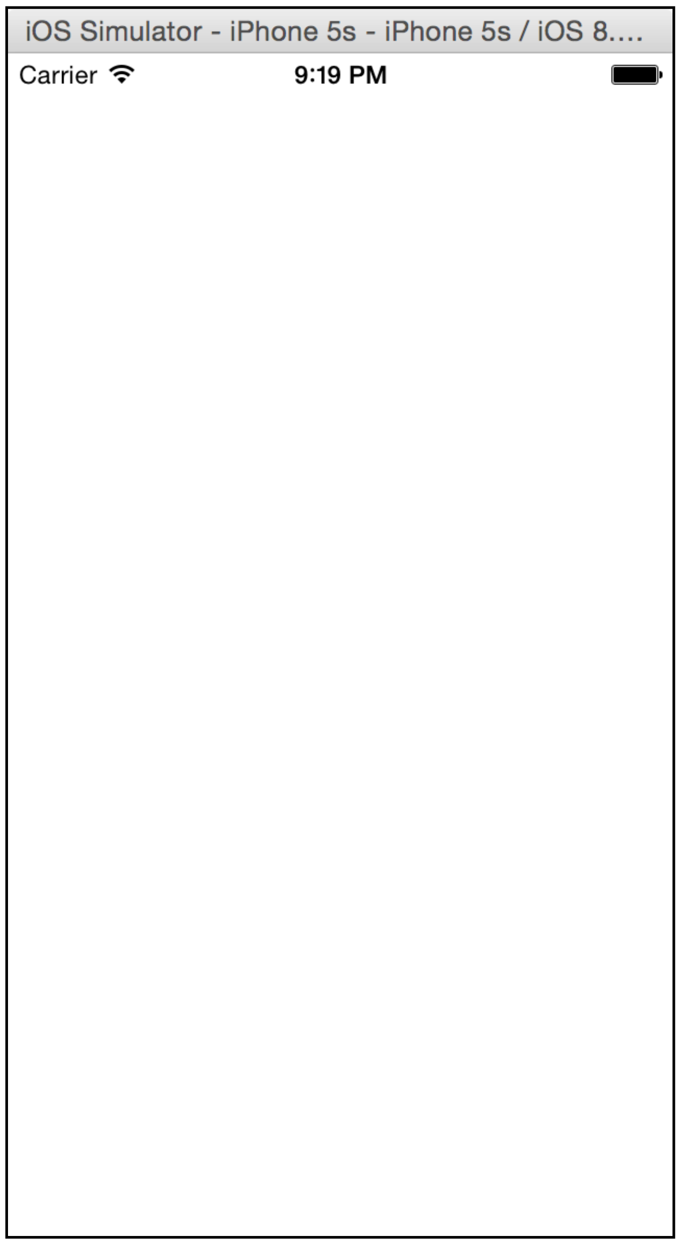


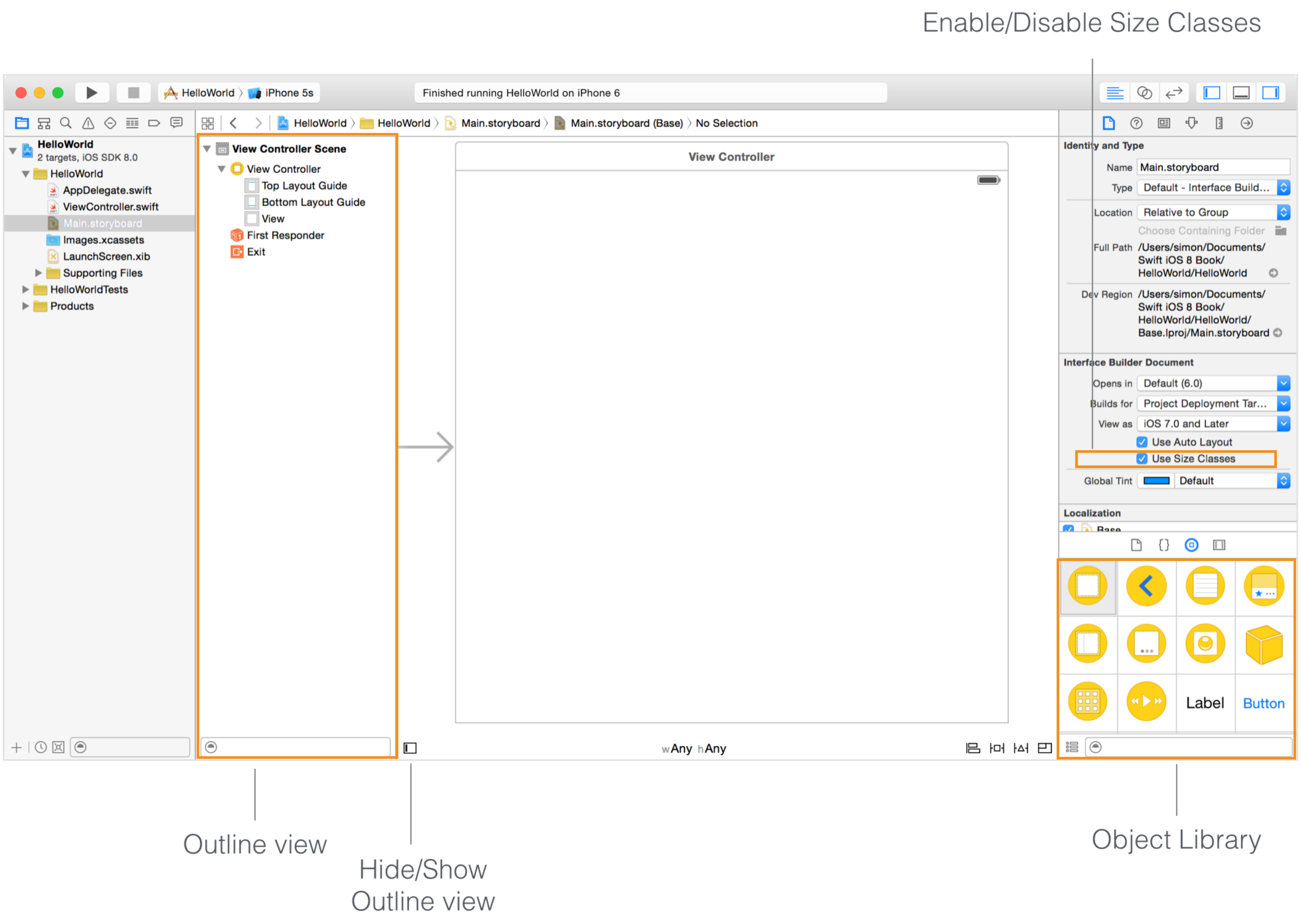
Figure 1-11. The Simulator

A white screen with nothing inside?! That's normal. Because we haven't implemented the user interface or written any lines of code, the Simulator shows a blank screen. To terminate the app, simply hit the "Stop" button in the toolbar.

Try to select another simulator and run the app. Just play around with it so you'll get used to the Xcode development environment.

Designing User Interface Using Storyboard

Now that you have a basic idea of the Xcode development environment, let's move on and design the user interface of your first app. In the project navigator, select the "Main.storyboard" file. Xcode then brings up a visual editor for Storyboards known as Interface Builder.



Identity and Type
Name: Main.storyboard
Type: Default - Interface Build...
Location: Relative to Group
Full Path: /Users/simon/Documents/Swift iOS 8 Book/HelloWorld/HelloWorld
Dev Region: /Users/simon/Documents/Swift iOS 8 Book/HelloWorld/HelloWorld/Base.lproj/Main.storyboard

Interface Builder Document
Opens in: Default (6.0)
Builds for: Project Deployment Tar...
View as: iOS 7.0 and Later
✓ Use Auto Layout
✓ Use Size Classes
Global Tint: Default

Localization
✓ Base

Figure 1-12 . Storyboard Editor

Storyboards provide a visual way for developers to create and design an app's UI. You use storyboard to lay out the views and the transitions between different views. Since we selected the "Single View Application" template, the storyboard already includes a view controller scene. A scene in storyboard represents a view controller and its views. When developing iOS apps, views are the basic building blocks for creating your user interface.

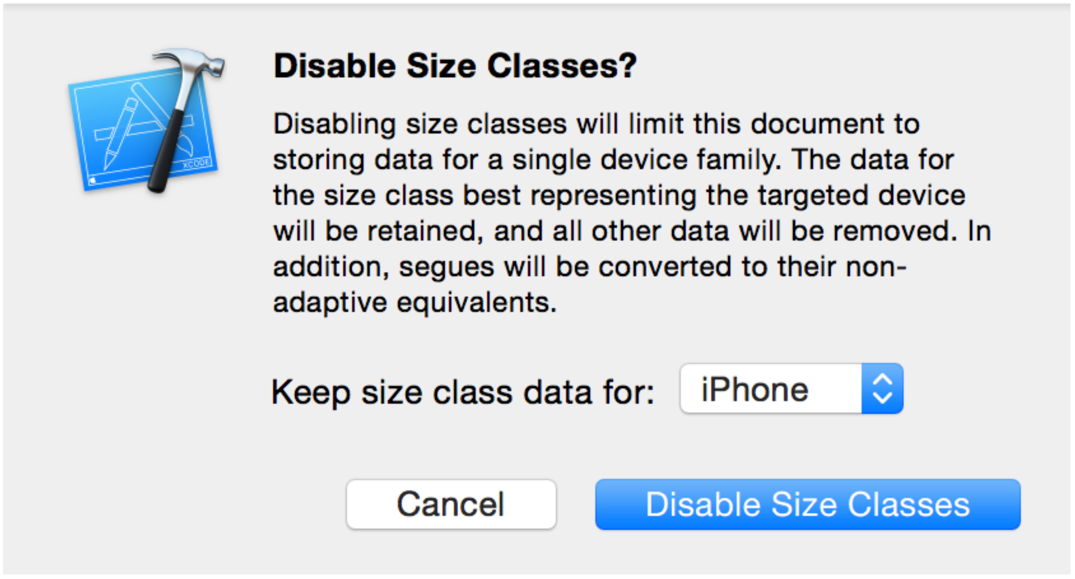
Each type of view has its own function. For instance, the view you find in the storyboard is a container view for holding other views such as buttons, labels, image views, etc.

A view controller is designed to manage its associated view and subviews (e.g. button and label). If you are confused about the relationship between views and view controllers, don't worry. We will discuss how a view and view controller work together in a later chapter. Meanwhile, focus on learning how to use storyboard and Interface Builder to lay out the UI.

The outline view of the Interface Builder shows you an overview of all scenes and the objects under a specific scene. The outline view is very useful when you want to select a particular object in the storyboard. If the outline view doesn't appear on screen, use the toggle button (see figure 1-12) to enable/disable the outline view.

Disabling Size Classes

If you have some experience with Xcode 5, you may wonder why the size of the view controller in Xcode 6 differs from the one in older version. The view controller is bigger and doesn't look like an iPhone. It's now a one-size-fits-all canvas. Why? This is due to the introduction of **Size Classes**.


Disable Size Classes?

Disabling size classes will limit this document to storing data for a single device family. The data for the size class best representing the targeted device will be retained, and all other data will be removed. In addition, segues will be converted to their non-adaptive equivalents.

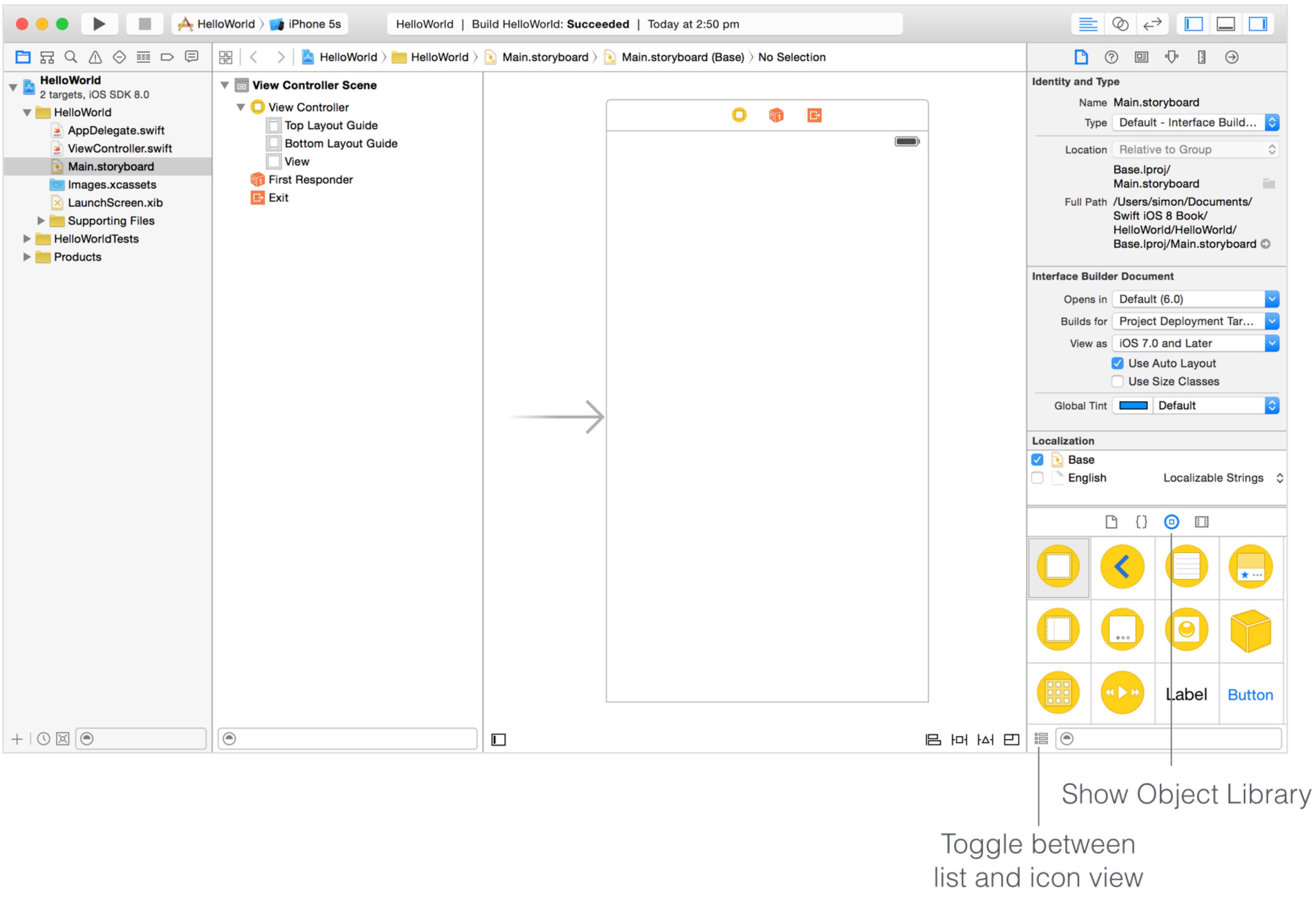
Keep size class data for: iPhone

Figure 1-13. Disable size classes

Size Classes are new classes introduced in iOS 8. By using size classes, Xcode 6 lets developers use a unified storyboard for creating an app UI that works well on both iPhone and iPad. Prior to that, if you needed to create a universal app that supports both iPad and iPhone, you'd need to create two different storyboards, one for each devices.

We'll not go into size classes here. To keep things simple, we'll disable size classes for your first project. In the File Inspector (see figure 1-12), uncheck the "Use Size Classes" checkbox under the Interface Builder Document. In case File Inspector is hidden, you can choose View > Utilities > Show File Inspector.

When you disable size classes, Xcode will prompt you to select the target device. For our project, select iPhone and click “Disable Size Classes” to confirm. The view controller now looks more like an iPhone.


HelloWorld
iPhone 5s
HelloWorld | Build HelloWorld: Succeeded | Today at 2:50 pm
HelloWorld > HelloWorld > Main.storyboard > Main.storyboard (Base) > No Selection
Show Object Library
Toggle between list and icon view

```
HelloWorld
  2 targets, iOS SDK 8.0
  HelloWorld
    AppDelegate.swift
    ViewController.swift
    Main.storyboard
    Images.xcassets
    LaunchScreen.xib
    Supporting Files
  HelloWorldTests
  Products
```

```
View Controller Scene
  View Controller
    Top Layout Guide
    Bottom Layout Guide
    View
  First Responder
  Exit
```

Identity and Type
Name Main.storyboard
Type Default - Interface Build...
Location Relative to Group
Base.lproj/Main.storyboard
Full Path /Users/simon/Documents/Swift iOS 8 Book/HelloWorld/HelloWorld/Base.lproj/Main.storyboard

Interface Builder Document
Opens in Default (6.0)
Builds for Project Deployment Tar...
View as iOS 7.0 and Later
☑ Use Auto Layout
☐ Use Size Classes
Global Tint Default

Localization
☑ Base
☐ English Localizable Strings

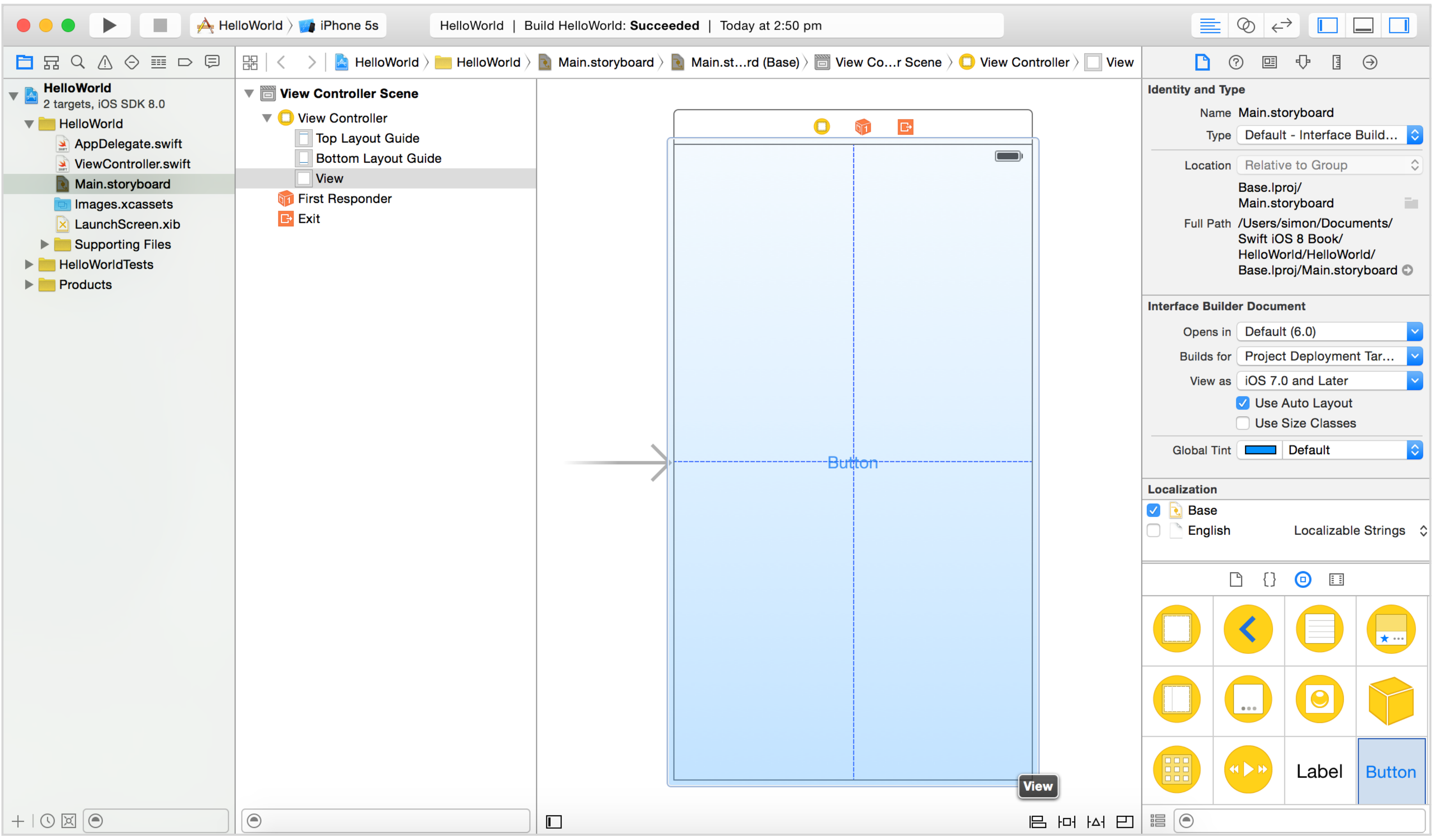
Figure 1-14. View Controller with size classes disabled

Adding a Button to the View

Next we’ll add a Hello World button to the view. At the bottom part of the utility area, it shows the Object library. Here, you can choose any of the UI Controls and drag-and-drop them into the view. If you don’t see the Object Library, you can click the “Show the Object Library” button.

You can use the toggle button to switch between list view and icon view (see figure 1-14). If you want to learn more about a specific object in the Object Library, simply click on it and Xcode shows you a brief description of the control.

Okay, it's time to add a button to the view. All you need to do is drag a Button object from the Object Library to the view.


HelloWorld | Build HelloWorld: **Succeeded** | Today at 2:50 pm

HelloWorld 〉 HelloWorld 〉 Main.storyboard 〉 Main.st…rd (Base) 〉 View Co…r Scene 〉 View Controller 〉 View

- HelloWorld
  2 targets, iOS SDK 8.0
  - HelloWorld
    - AppDelegate.swift
    - ViewController.swift
    - Main.storyboard
    - Images.xcassets
    - LaunchScreen.xib
    - Supporting Files
  - HelloWorldTests
  - Products

- View Controller Scene
  - View Controller
    - Top Layout Guide
    - Bottom Layout Guide
    - View
  - First Responder
  - Exit

Identity and Type
Name: Main.storyboard
Type: Default - Interface Build…
Location: Relative to Group
Base.lproj/Main.storyboard
Full Path: /Users/simon/Documents/Swift iOS 8 Book/HelloWorld/HelloWorld/Base.lproj/Main.storyboard

Interface Builder Document
Opens in: Default (6.0)
Builds for: Project Deployment Tar…
View as: iOS 7.0 and Later
☑ Use Auto Layout
☐ Use Size Classes
Global Tint: Default

Localization
☑ Base
☐ English — Localizable Strings

Figure 1-15. Drag the Button to the View

As you drag the Button to the view, you'll see a set of horizontal and vertical guides if the button is centered. Stop dragging, and release your button to place the Button object there.

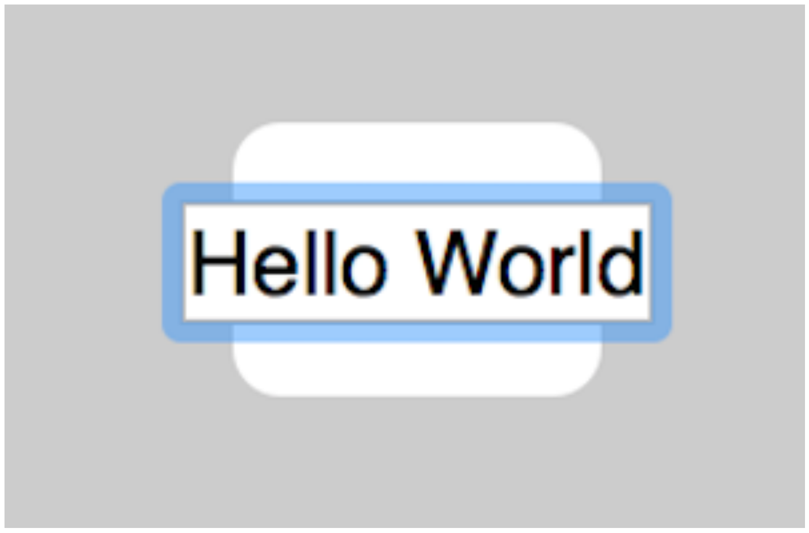


Figure 1-16. Renaming the button

Next, let's rename the button. To edit the label of the button, double-click it and name it "Hello World".

If you hit the Run button to run the app, you'll see a Hello World button in the simulator as shown in figure 1-17. Cool, right? However, when you tap the button, it does nothing. We'll need to add a few lines of code to display the "Hello, World" message.

This is the beauty of iOS development. The code and user interface of an app are separated. You're free to design your user interface in Storyboard and prototype an app without writing any lines of code.

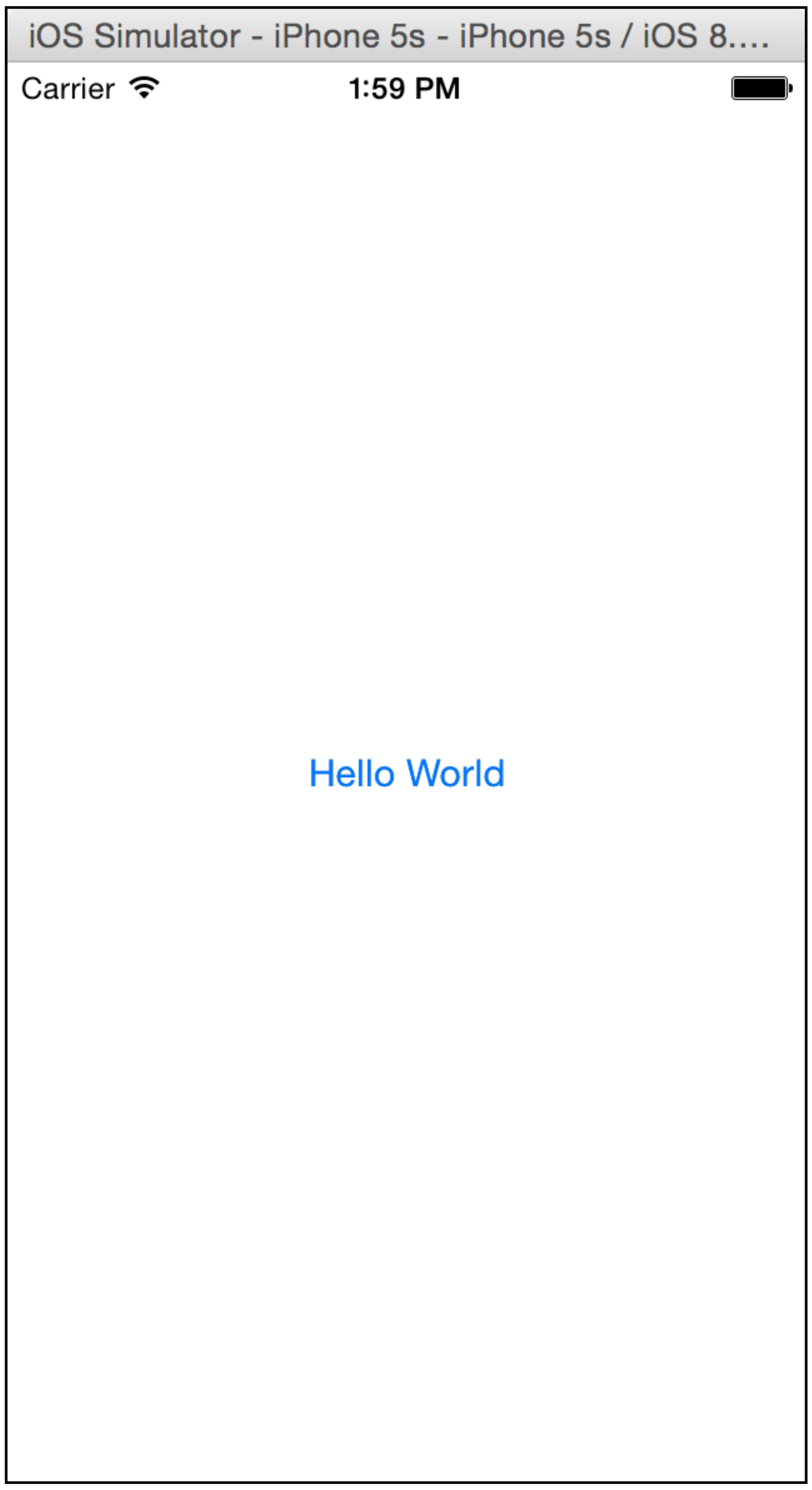


Figure 1-17. Hello World app with a Button

Coding the Hello World Button

Now that you've completed the UI of the HelloWorld app, it's time to write some code. In the Project Navigator, you should find the *ViewController.swift* file. Because we initially selected the "Single View Application" project template, Xcode already generated a ViewController class in the *ViewController.swift*. In order to display a message when the button is tapped, we'll add some code to the file.

Swift versus Objective-C

If you have written code in Objective-C before, one big change in Swift is the consolidation of header (.h) and implementation file (.m). All the information of a particular class is now stored in a single .swift file.

Select the file and the editor area immediately displays the source code. Type (I encourage you to type the code, rather than copy & paste) the following lines of code in the *ViewController* class:

```
@IBAction func showMessage() {
     let alertController = UIAlertController(title: "Welcome to My First App", message: "Hello
World", preferredStyle: UIAlertControllerStyle.Alert)
     alertController.addAction(UIAlertAction(title: "OK", style: UIAlertActionStyle.Default,
handler: nil))
     self.presentViewController(alertController, animated: true, completion: nil)

}
```

Your source code should look like this after editing:

```
import UIKit

class ViewController: UIViewController {

    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.
    }

    @IBAction func showMessage() {
        let alertController = UIAlertController(title: "Welcome to My First App", message:
"Hello World", preferredStyle: UIAlertControllerStyle.Alert)
        alertController.addAction(UIAlertAction(title: "OK", style: UIAlertActionStyle.Default,
handler: nil))
        self.presentViewController(alertController, animated: true, completion: nil)

    }
}
```

What you have just done is added a *showMessage()* method in the *ViewController* class. The Swift code within the method is new to you. I will explain it to you in the next chapter. Meanwhile, just consider the *showMessage()* as an action. When this action is called, the block of code will instruct iOS to display a “Hello World” message on screen.

Connecting User Interface with Code

I said before that the beauty of iOS development is the separation of code (.swift file) and user interface (storyboards). But how can we establish the relationship between our source code and user interface?

To be specific for this demo, the question is:

How can we connect the "Hello World" button in the storyboard with the showMessage() method in the ViewController class?

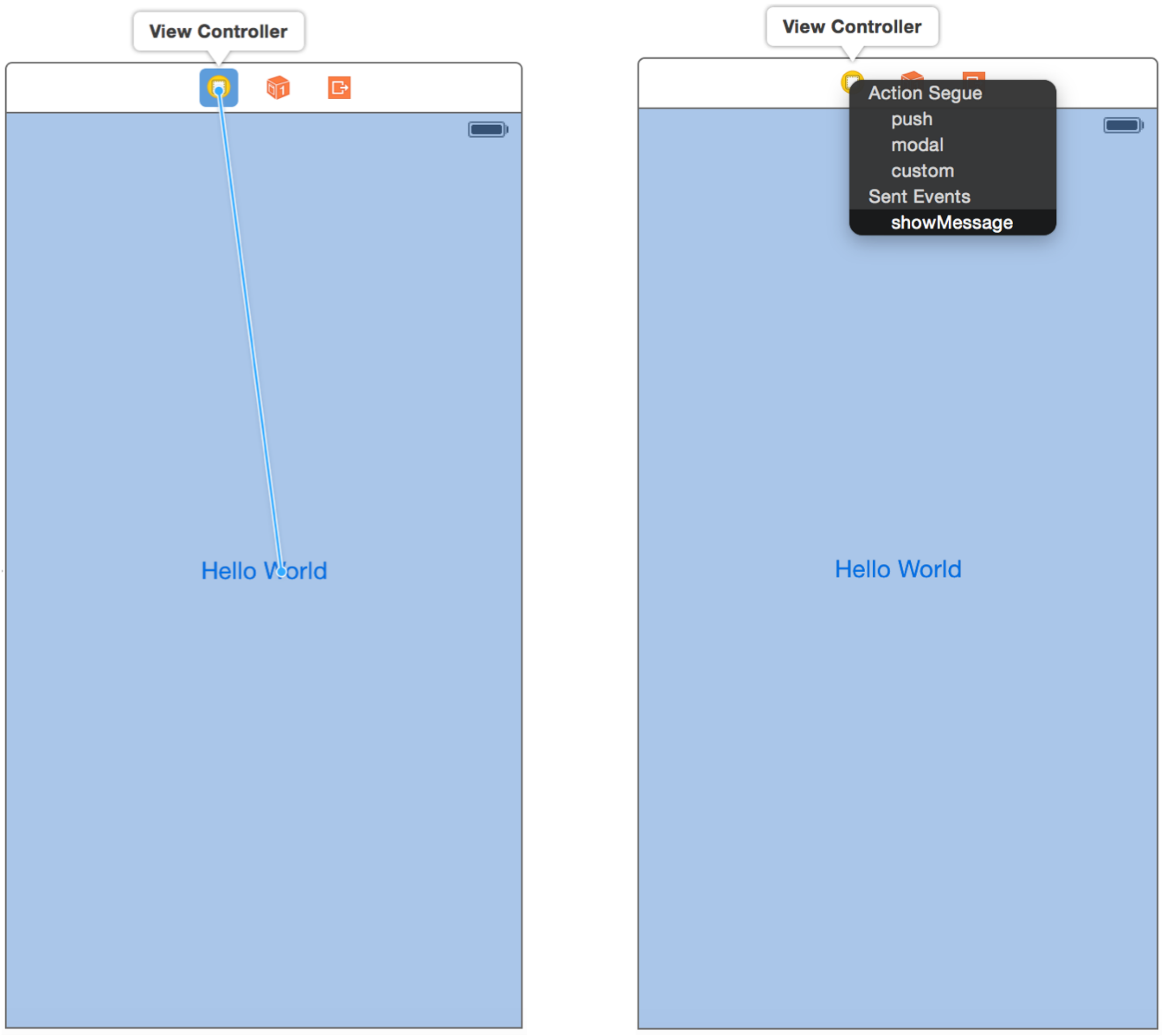


Figure 1-18. Drag to the View Controller icon (left), a pop-over menu appears when releasing the buttons (right)

You need to establish a connection between the "Hello World" button and the *showMessage()* method you've just added, so that the app responds when someone taps the Hello World button. Select the "Main.storyboard" to switch back to the Interface Builder.

Press and hold the *control* key on your keyboard, click the "Hello World" button and drag it to the View Controller icon.

Release both buttons (mouse + keyboard) and a pop-up shows the "showMessage" option under Sent Events. Select it to make a connection between the button and "showMessage" action.

Test Your App

That's it! You're now ready to test your first app. Just hit the "Run" button. If everything is correct, your app should run properly in the Simulator. This time, the app displays a welcome message when you tap the Hello World button.

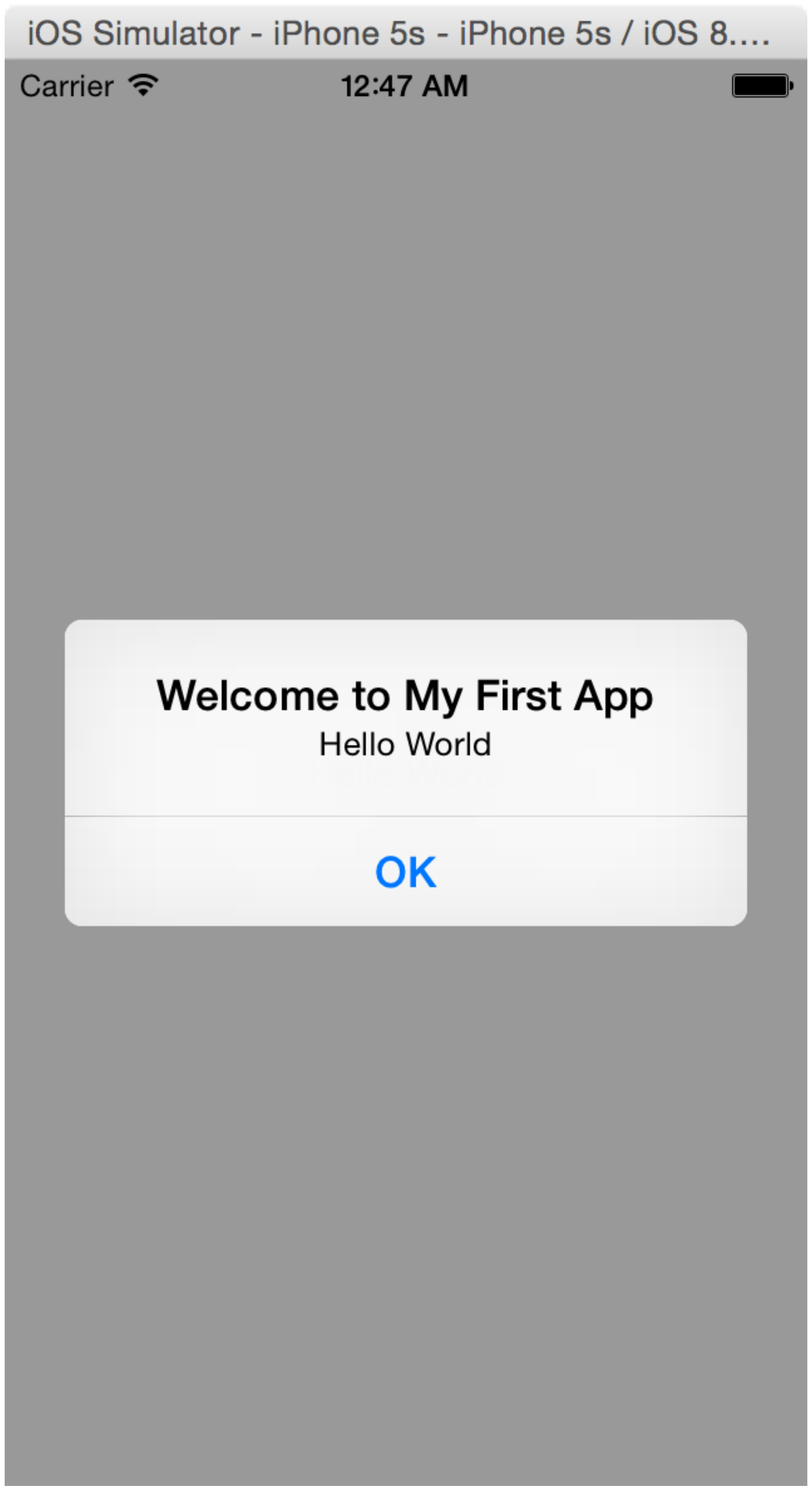
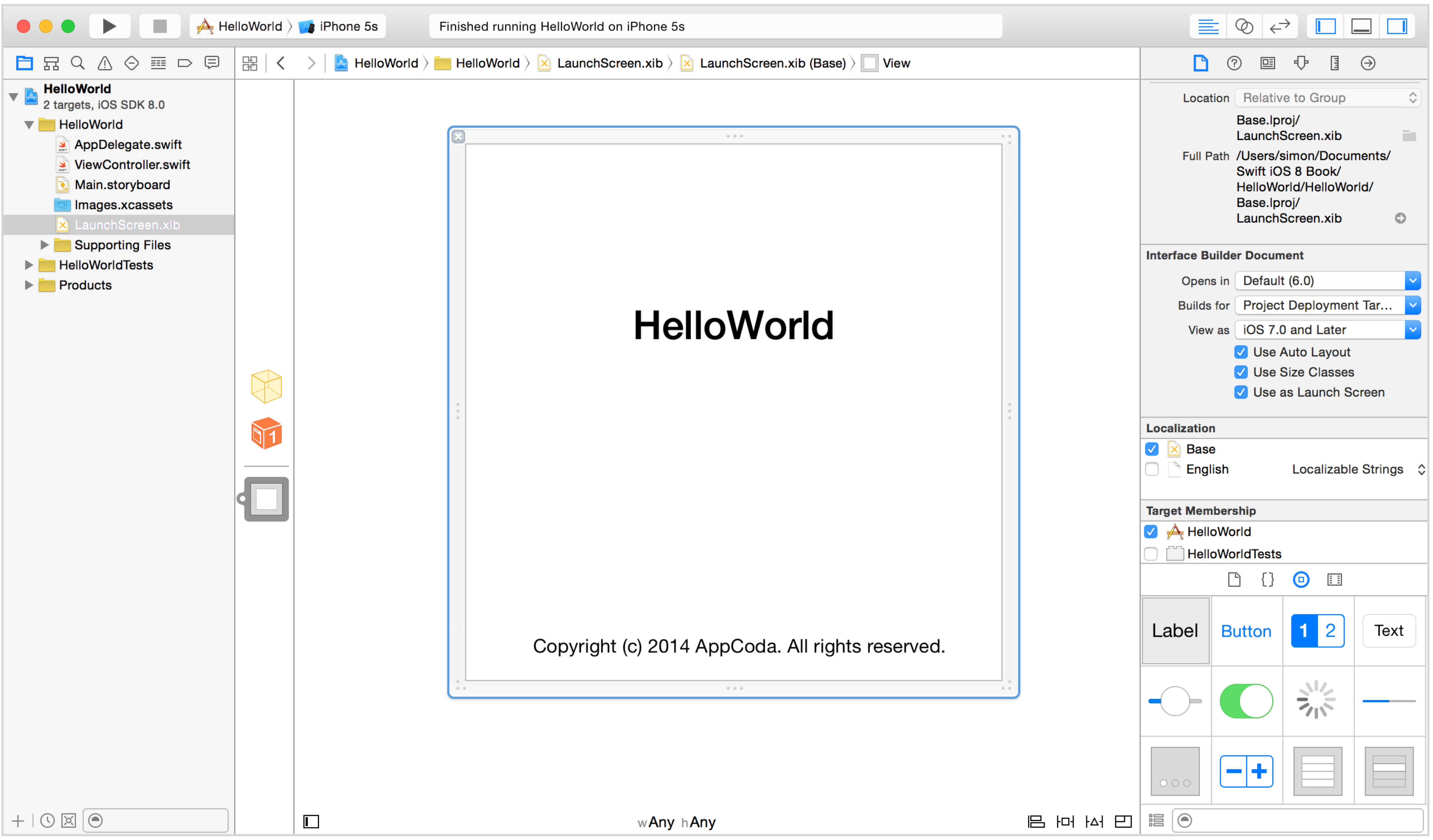


Figure 1-19. Hello World app

Launch Screen

When the app starts up, you may aware a launch screen as shown in figure 1-20. The screen disappears when the Hello World screen displays.



Location: Relative to Group
Base.lproj/LaunchScreen.xib
Full Path: /Users/simon/Documents/Swift iOS 8 Book/HelloWorld/HelloWorld/Base.lproj/LaunchScreen.xib

Interface Builder Document
Opens in: Default (6.0)
Builds for: Project Deployment Tar...
View as: iOS 7.0 and Later
☑ Use Auto Layout
☑ Use Size Classes
☑ Use as Launch Screen

Localization
☑ Base
☐ English — Localizable Strings

Target Membership
☑ HelloWorld
☐ HelloWorldTests

Figure 1-20. Launch screen

Traditionally this has been a static image (we called it launch image) displayed immediately on app launch, before the actual app UI is ready to go. The launch image gives users the impression that your app is fast and responsive as it appears instantly. In iOS 8 and Xcode 6, Apple allows developer to create the launch screen using Interface Builder instead of a static launch image.

For any new projects created in Xcode 6, you will find a XIB launch screen file, configured as the default launch file. Like the HelloWorld project, you should find the *LaunchScreen.xib* in the project navigator. By default, the screen contains the name of your project and the copyright notice. You can customize the screen just like you design Hello World view controller. Meanwhile, you can leave it as it is. We'll talk about launch image again when you finish building a real app.

Changing the Button Color

There is one more thing I want to discuss with you before ending the chapter. As mentioned before, you do not need to write code to customize a UI control. Here, I want to show you how easy it is to change the properties (e.g. color) of a button. Select the “Hello World” button and then click the Attributes Inspector under the Utility area. You’ll be able to access the properties of the button. Here, you can change the font, text color, background color, etc. Try to change the text color (under Button section) to *white* and background (scroll down and you’ll find it under View section) to orange or whatever color you want.

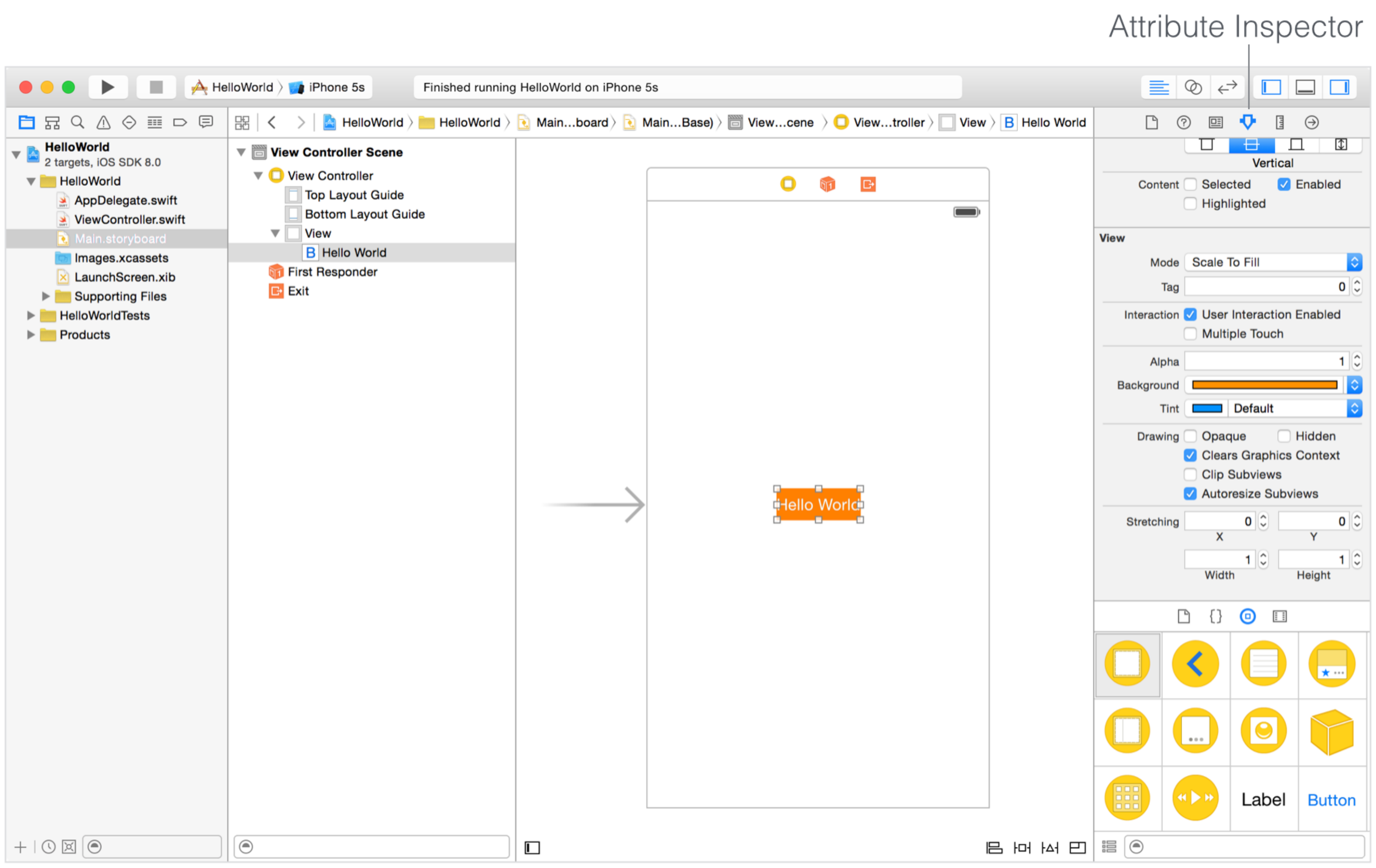


Figure 1-21. Changing the color of the Hello World button

What’s Coming Next

Congratulations! You’ve built your first iPhone app. It’s a simple app, but I believe you already have a better understanding of Xcode 6 and understand how an app is built. It’s easier than you thought, right?

In the next chapter, we’ll discuss the details of the Hello World app and explain how everything works together.

For your reference, you can download the complete Xcode project from https://www.dropbox.com/s/uiy4b31da10k58c/HelloWorld.zip