Chapter 3
Hello World! Build Your First App Using Swift

> Learn by doing. Theory is nice but nothing replaces actual experience.
>
> – Tony Hsieh

By now you should have installed Xcode 7 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 7.0 (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.

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 of the syntax and structure of Swift, the new programming language of iOS.
- It also gives you a basic introduction to the Xcode 7 environment. You'll learn how to create an Xcode project and lay out your user interface using Interface Builder. Even if you've used Xcode 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 built-in 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 3-1, 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.

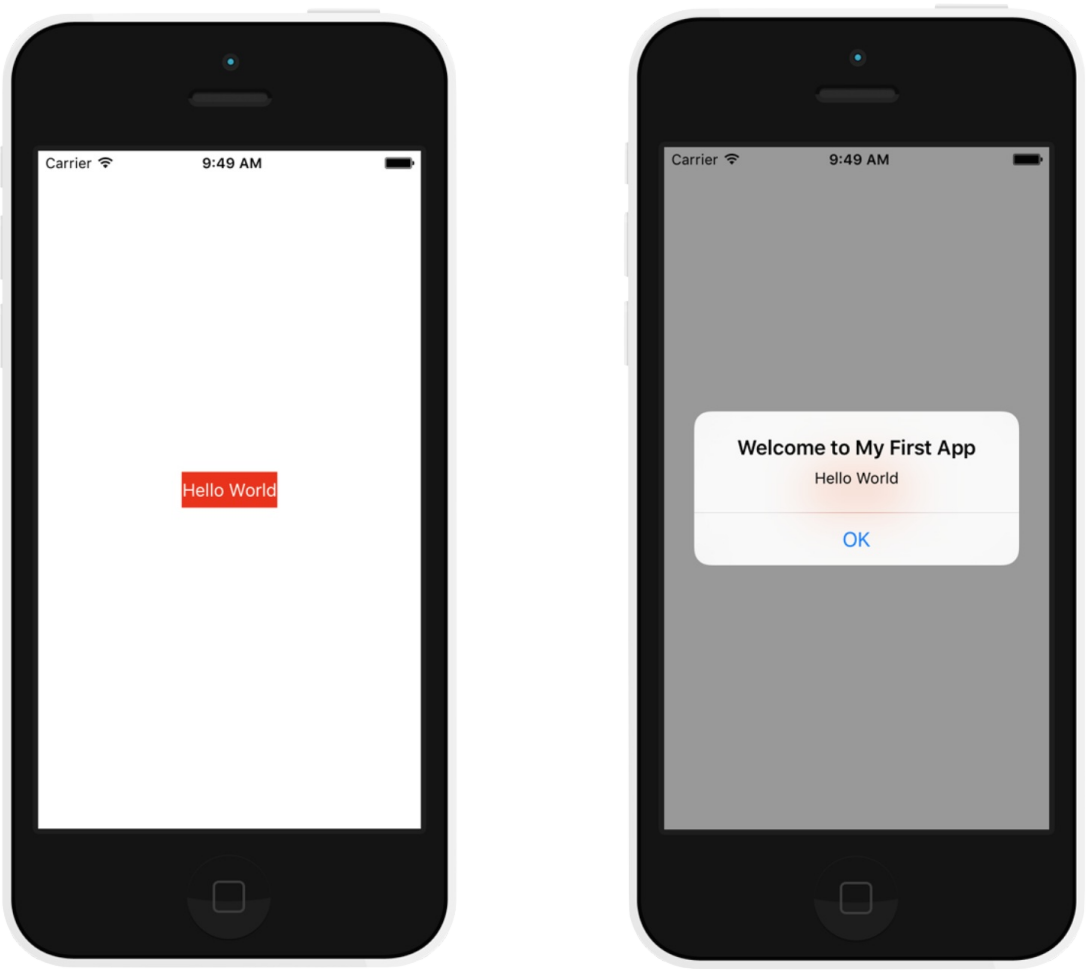

Figure 3-1. HelloWorld App

Let's Jump Right Into Create a Project

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

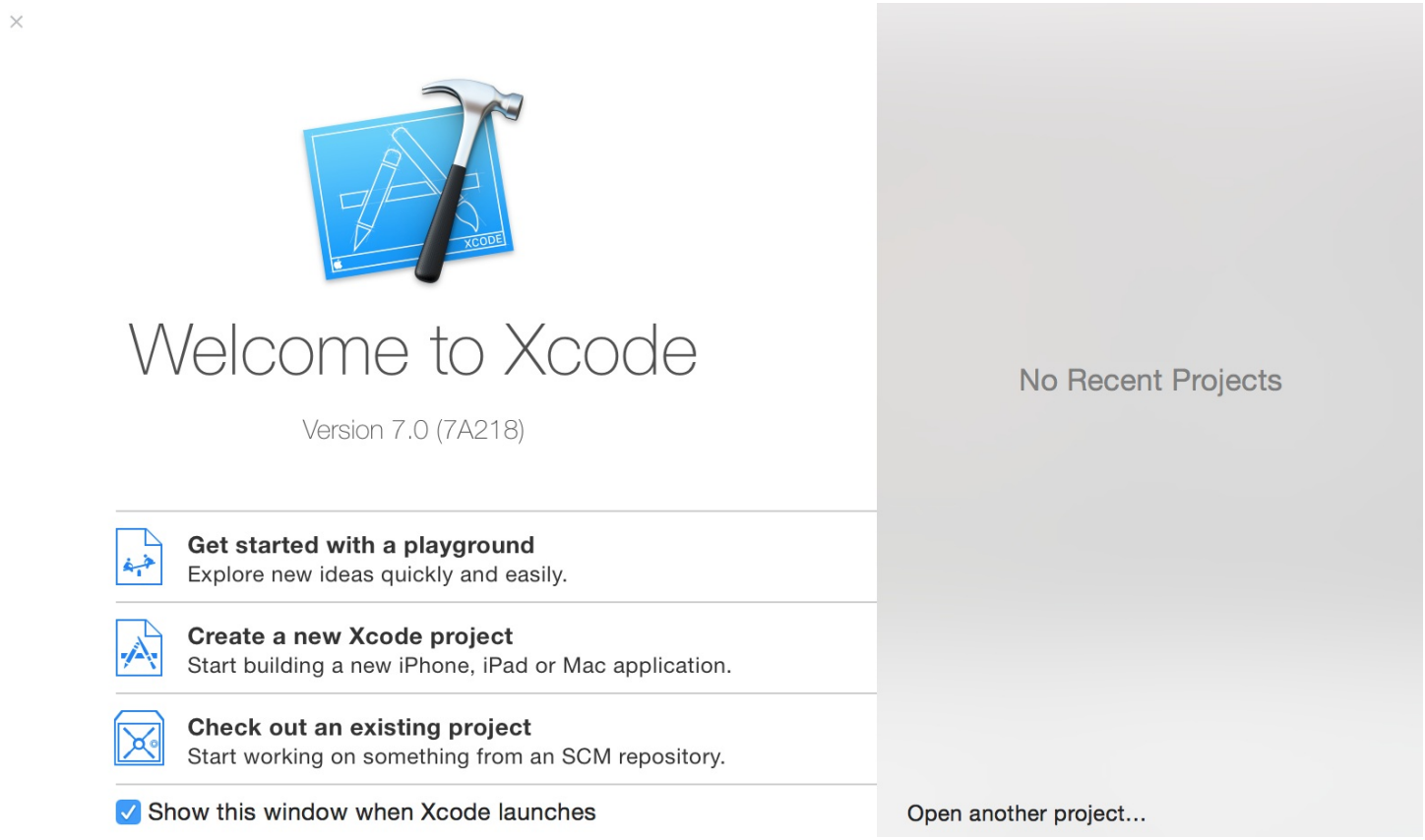

Figure 3-2. Xcode - Welcome Dialog

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

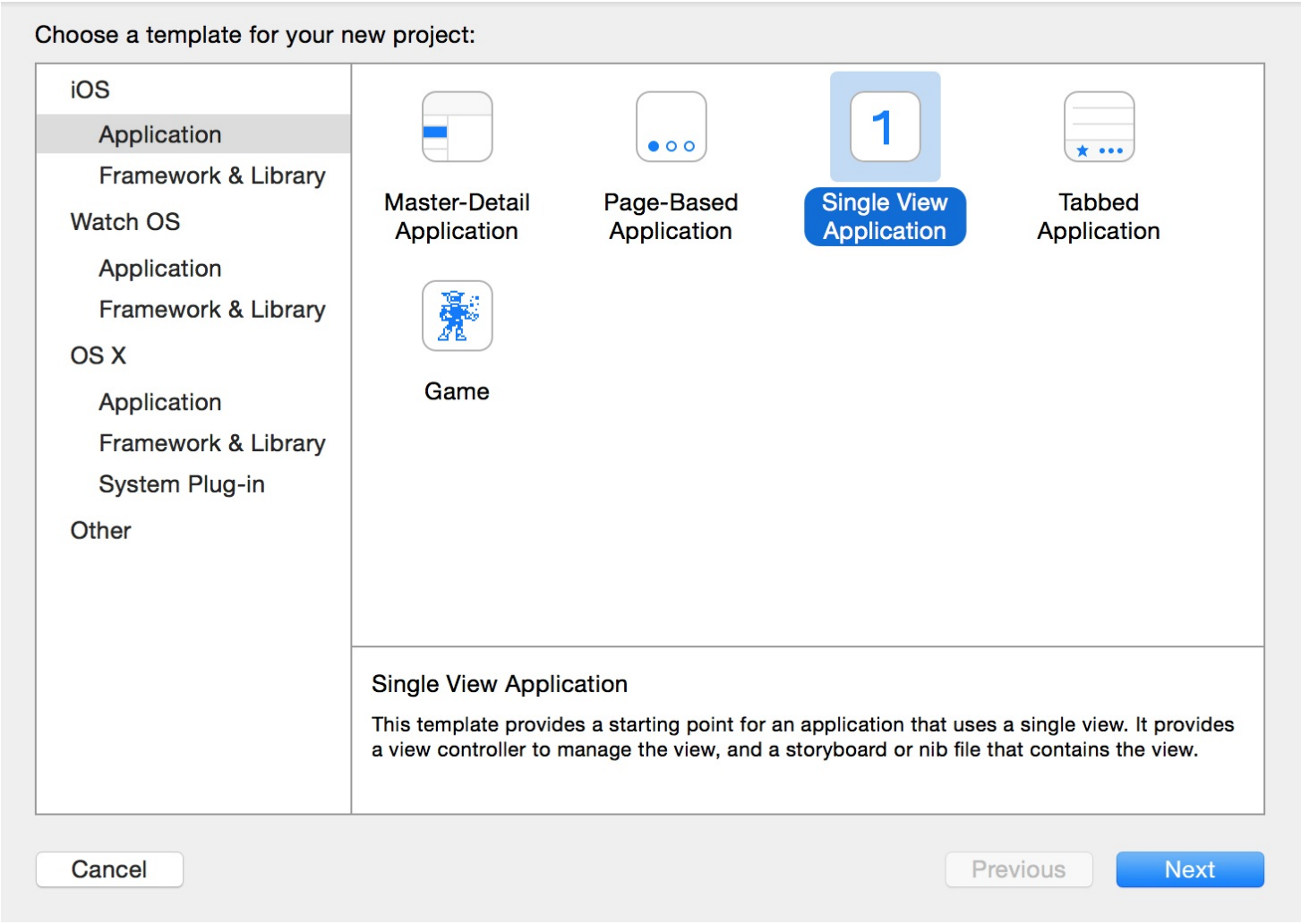

Figure 3-3. Xcode Project Template Selection

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

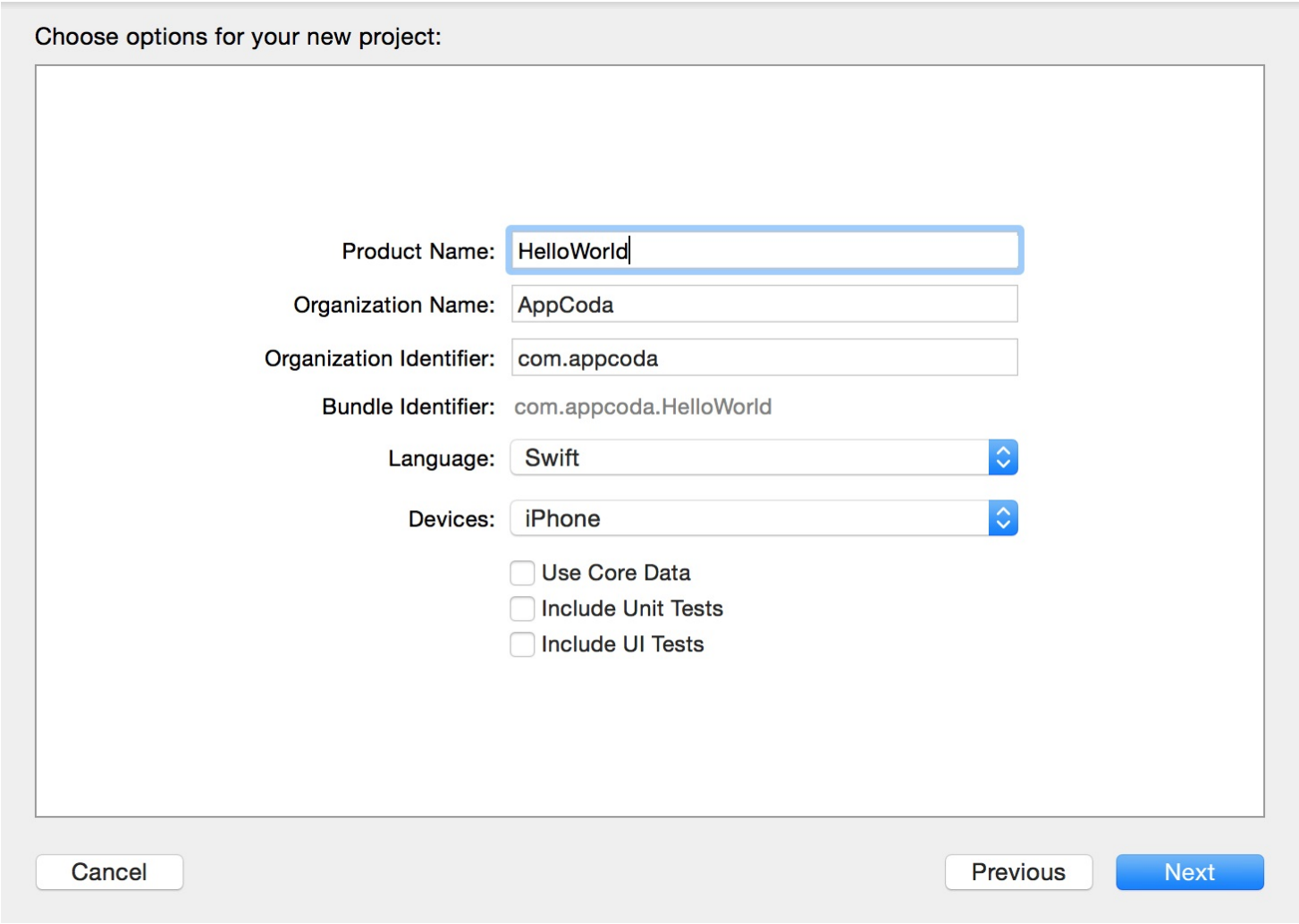

Figure 3-4. Options for your Hello World project

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 7 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.
- **Include Unit Tests: [unchecked]** – Do not select this option. You do not need unit tests for this simple project.
- **Include UI Tests: [unchecked]** – Do not select this option. You do not need UI tests for this simple 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.

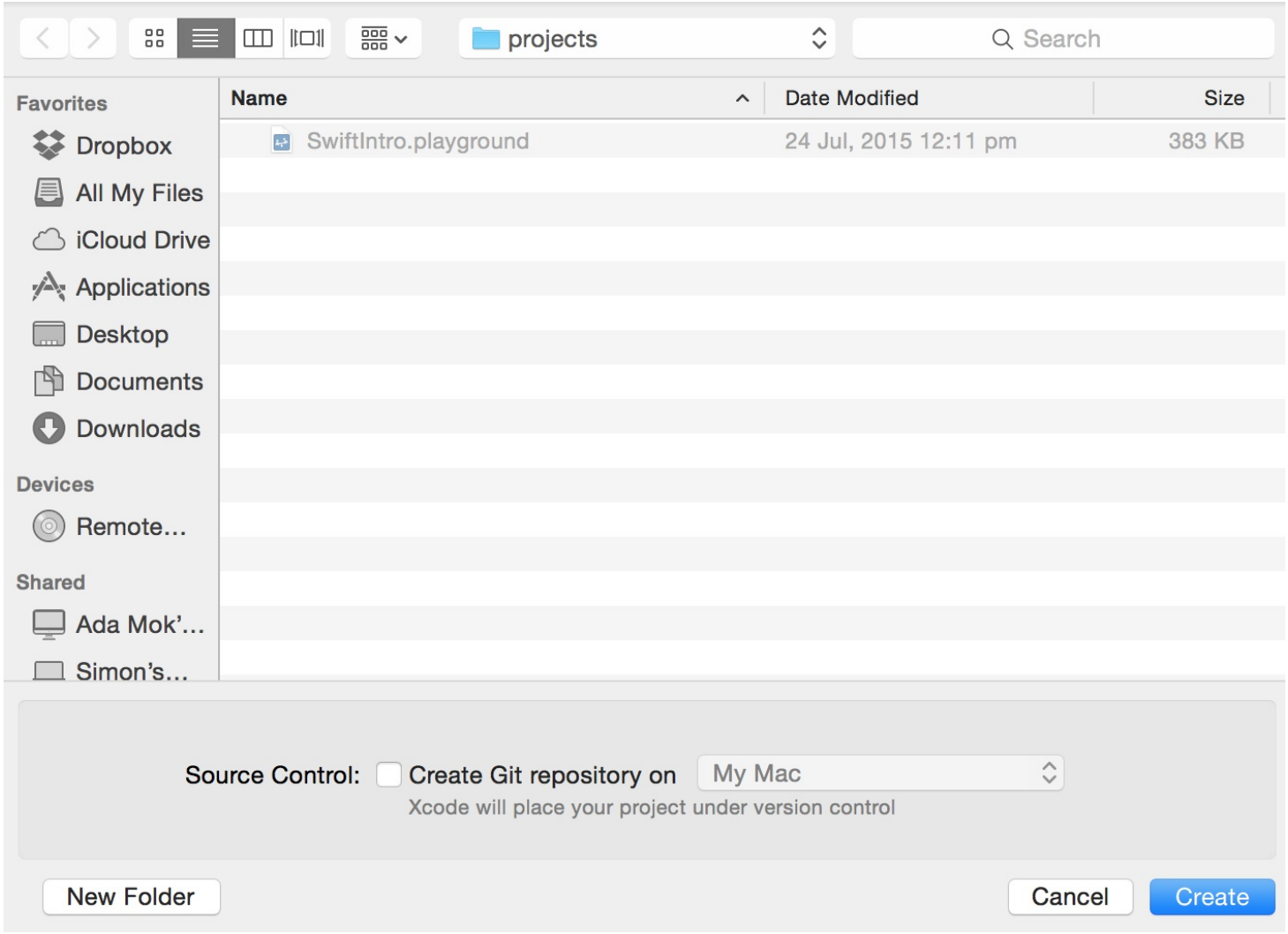

Figure 3-5. Choose a folder and save your project

After you confirm, Xcode automatically creates the "Hello World" project. The screen will look like the screenshot shown in figure 3-6. In case Xcode display the "No matching signing identity found" error under the Team option, you can just ignore it.

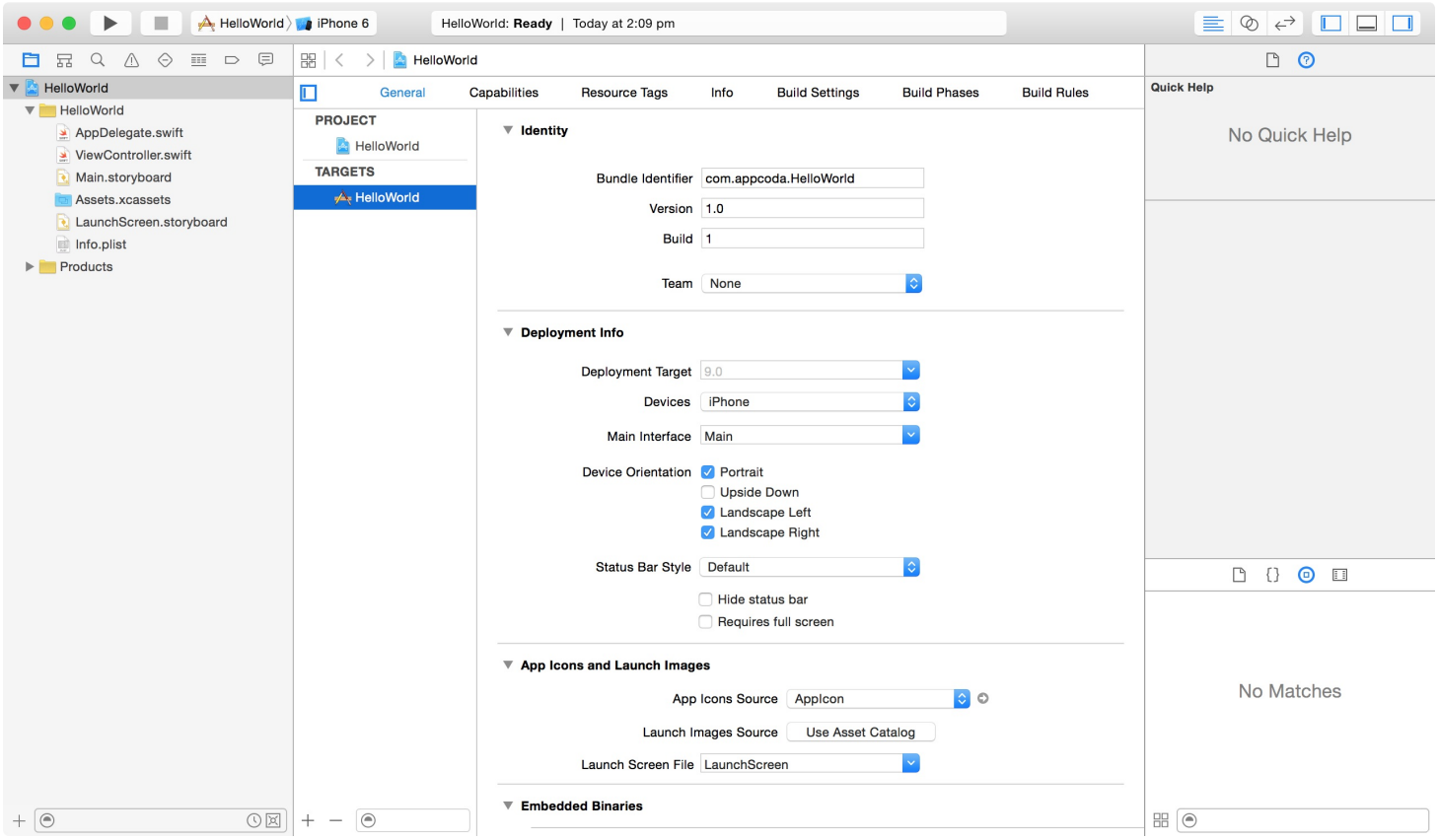

Figure 3-6. Main Xcode Window for HelloWorld Project

Quick note: As you go through the chapter, 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 Interface Builder. I will explain the code as we go along and you will learn how the HelloWorld app works in the next chapter.

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 3-7). If you select Main.storyboard, which is the file for storing user interface, Xcode shows you the visual editor for storyboard (see figure 3-8).

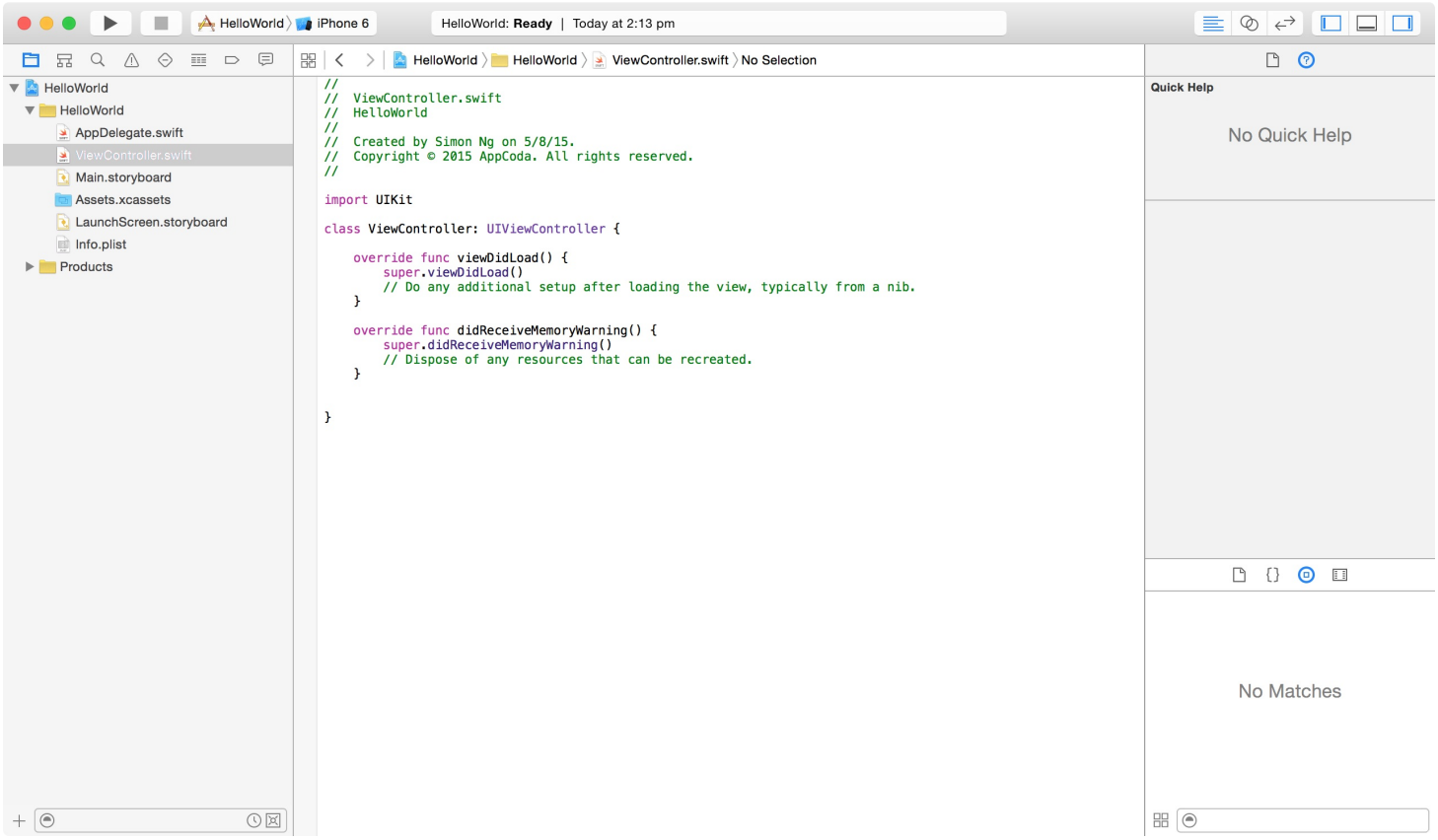

Figure 3-7. Xcode Workspace with Source Code Editor

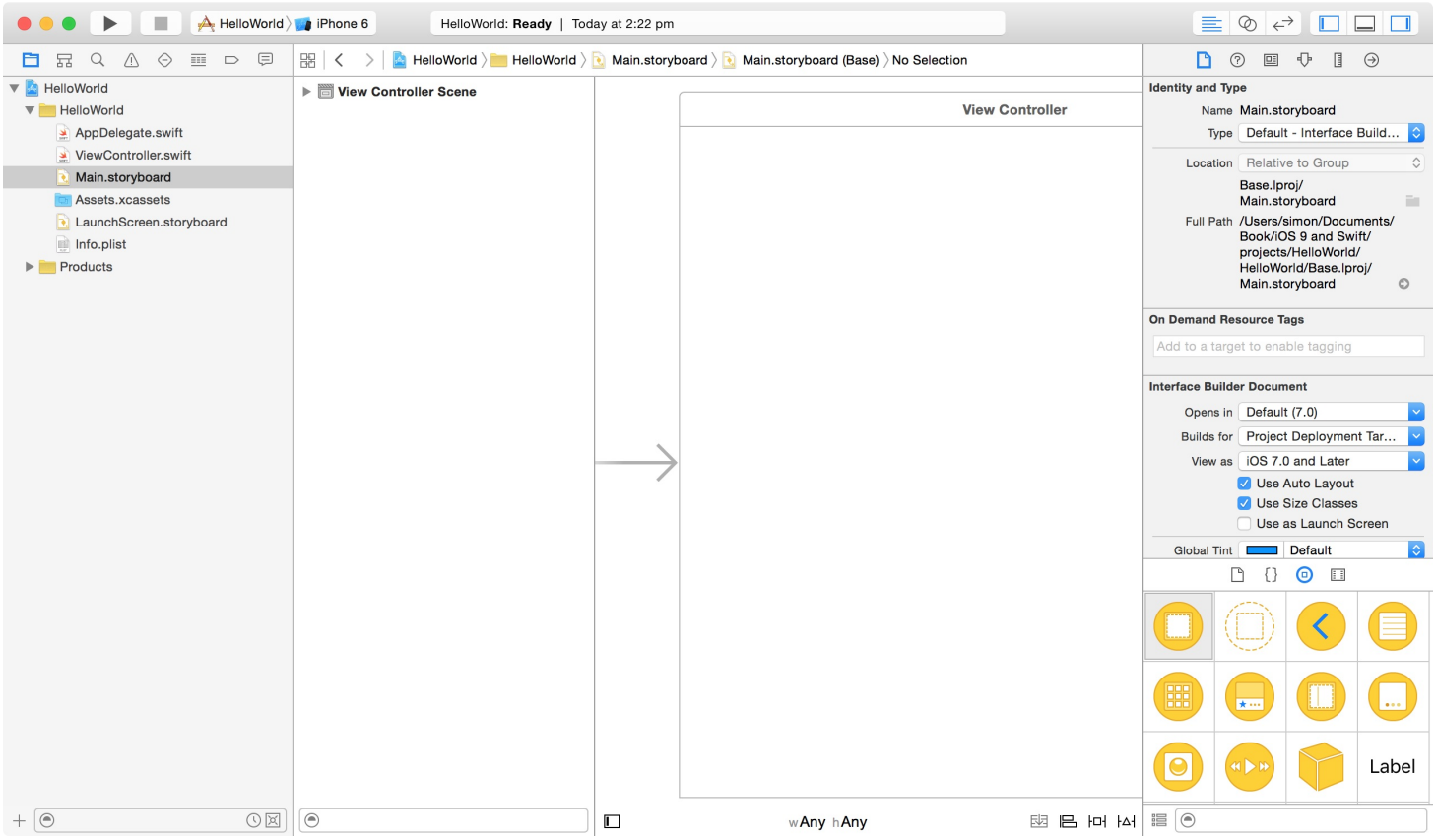

Figure 3-8 . Xcode Workspace with Storyboard Editor

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.

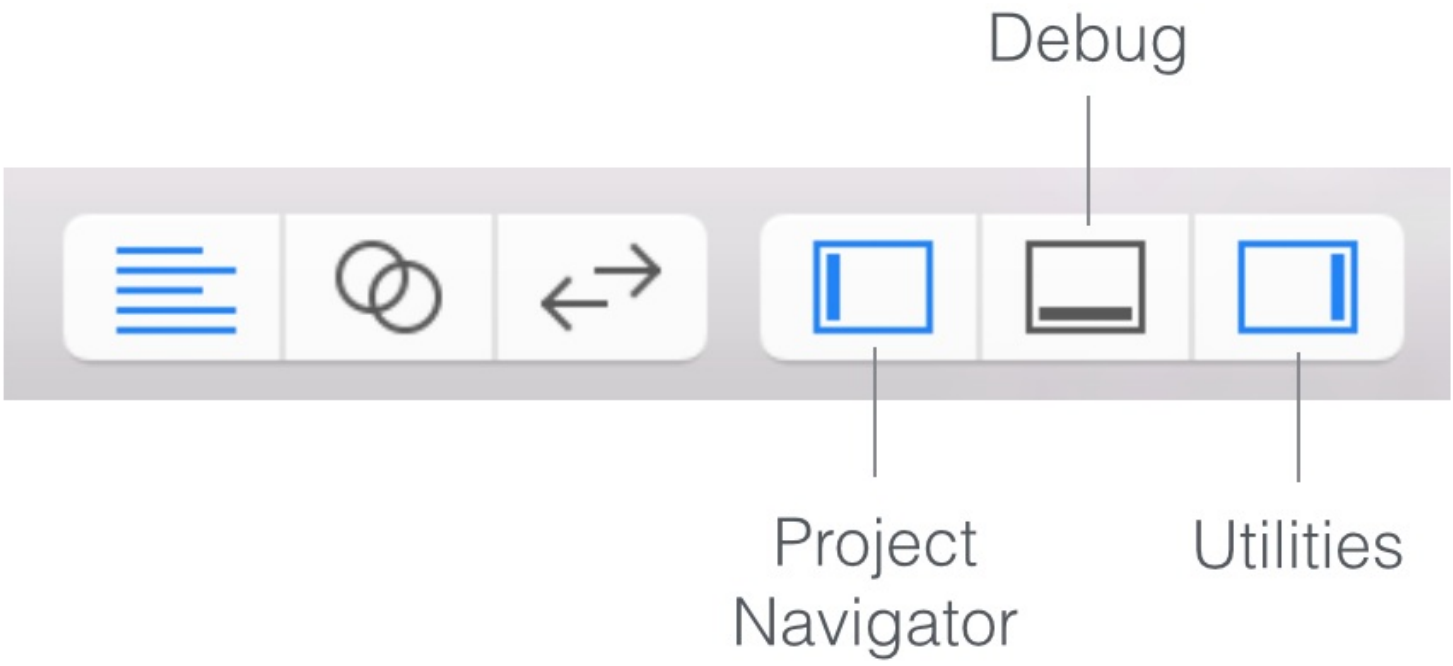

Figure 3-9. Show/hide the content areas of your workspace

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.

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 such as iPhone 4s and iPhone 6 Plus. As we're going to build an iPhone app, we simply use iPhone 6 as the Simulator.

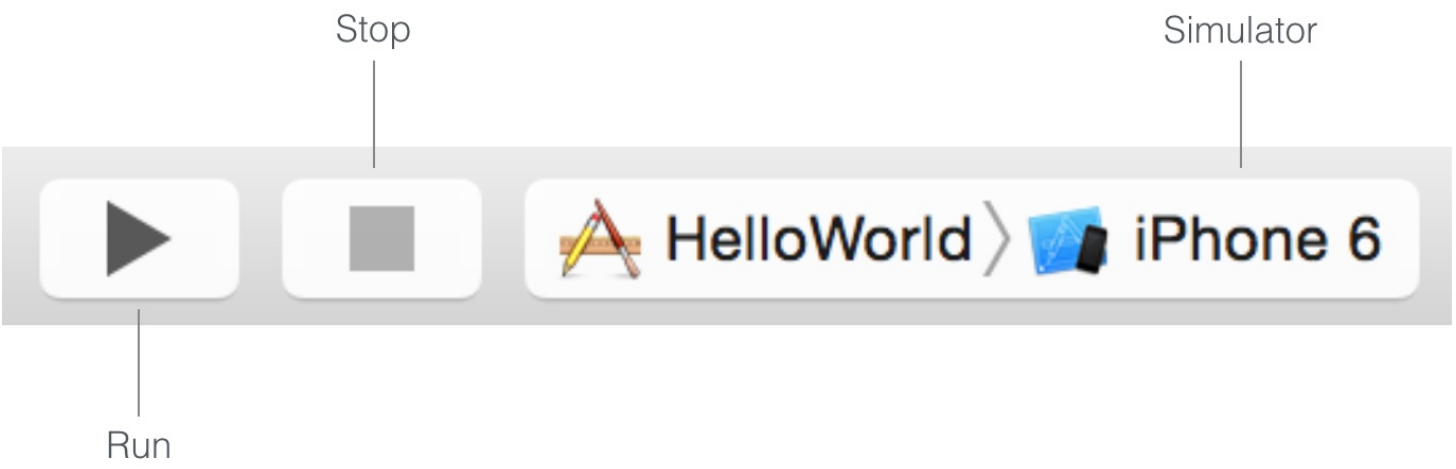

Figure 3-10. Run and Stop Buttons in Xcode

Once selected, you can click the Run button to load your app in the Simulator. Figure 3-11 shows the simulator of an iPhone 6.

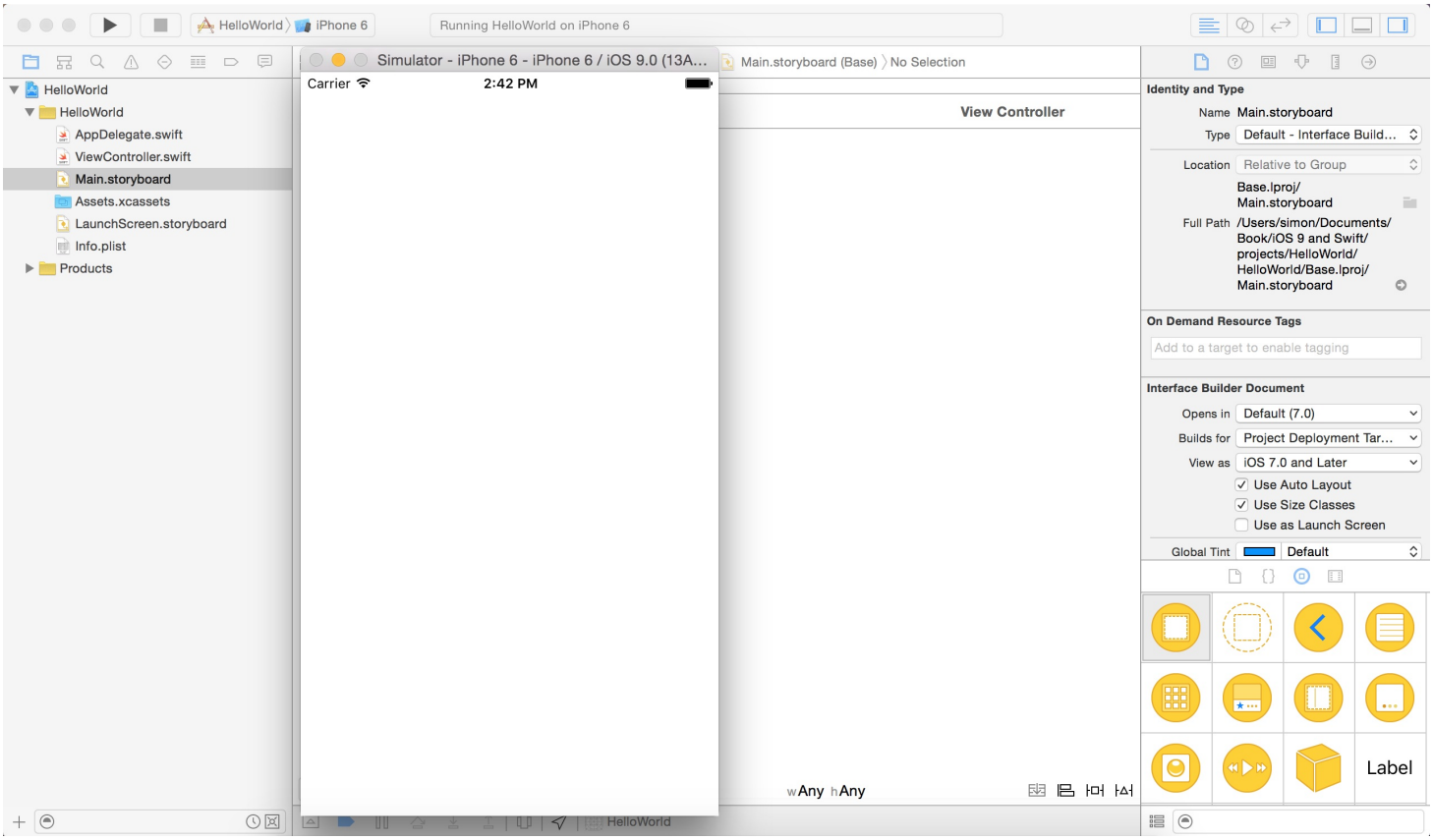

Figure 3-11. The Simulator

A white screen with nothing inside?! That's normal. So far we haven't designed the user interface or written any lines of code. This is why the simulator shows a blank screen. To terminate the app, simply hit the "Stop" button in the toolbar.

```
Quick tip: On non-retina Mac, it may not be able to show the full simulator
window. You can select the simulator, and press command+2 or command+3 to scale
it down. Alternatively, you can go up to the menu, and choose Window > Scale.
```

Try to select another simulator (e.g. iPhone 5) and run the app. Just play around with it so you'll get used to the Xcode development environment.

Designing User Interfaces 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.

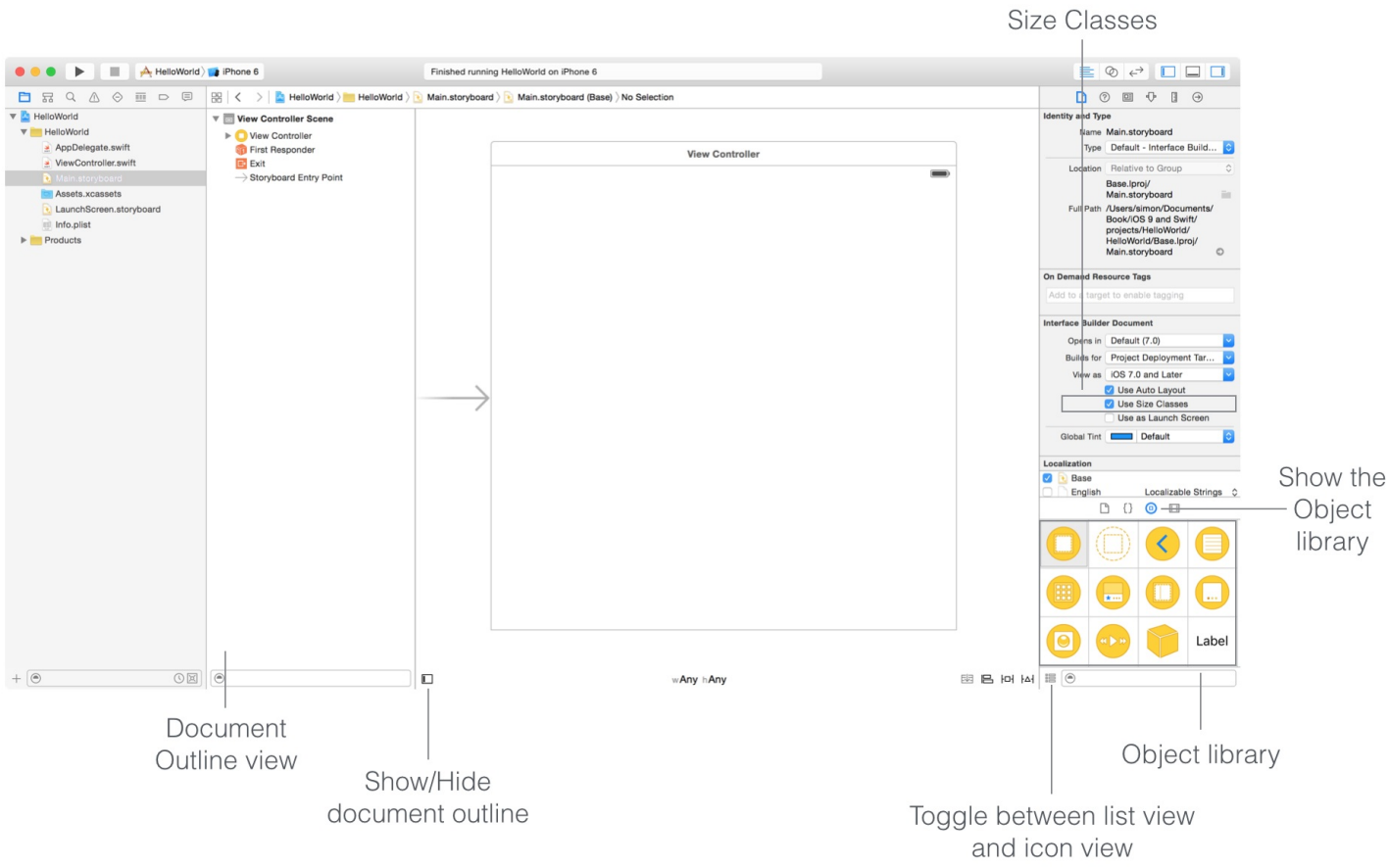

Figure 3-12 . The Interface Builder Editor

The Interface Builder editor provides a visual way for developers to create and design an app's UI. Not only can you use it to design individual view, the Interface Builder's storyboard designer lets you lay out multiple views, and chain them together using different types of transitions to create the complete user interface. All these can be done without writing a line of code.

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 the later chapters. Meanwhile, focus on learning how to use Interface Builder to lay out the UI.

The Document Outline view of the Interface Builder editor 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 3-12) 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 6/7 differs from that in the older version. The view controller is bigger and doesn't look like an iPhone. It's now a one-size-fits-all canvas. Why is that? This is due to the introduction of Size Classes.

Size Classes are new classes, which were first introduced in iOS 8. With size classes, developers can use a unified storyboard to create app UIs that work on both iPhone and iPad. Prior to that, if you needed to create a universal app that supports both types of devices, you would

need to create two different storyboards, one for each device type.

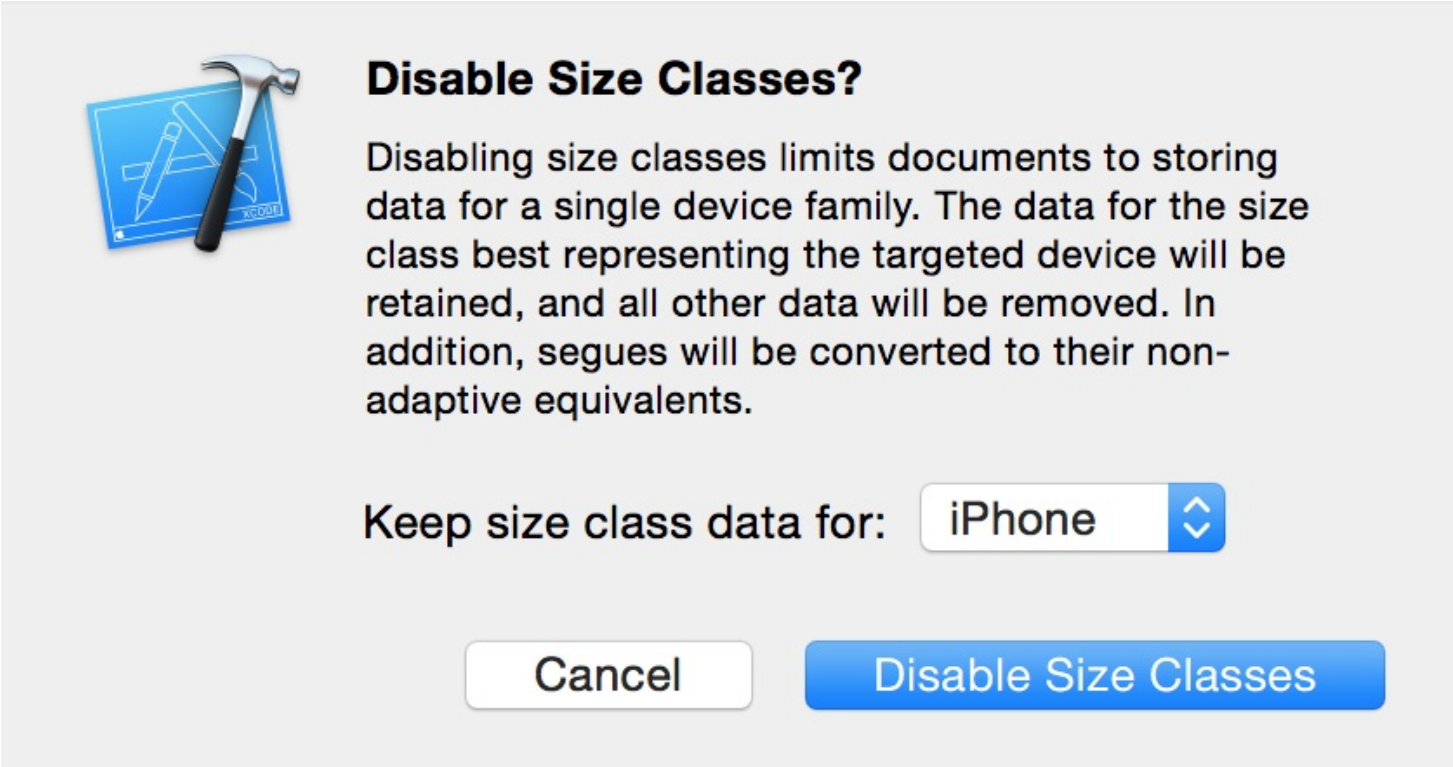

Figure 3-13. Disable size classes

We'll not go into size classes in this chapter. To keep things simple, we'll disable size classes for your first project. In the File inspector (see figure 3-12), uncheck the "Use Size Classes" checkbox under the Interface Builder Document. In case the 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.

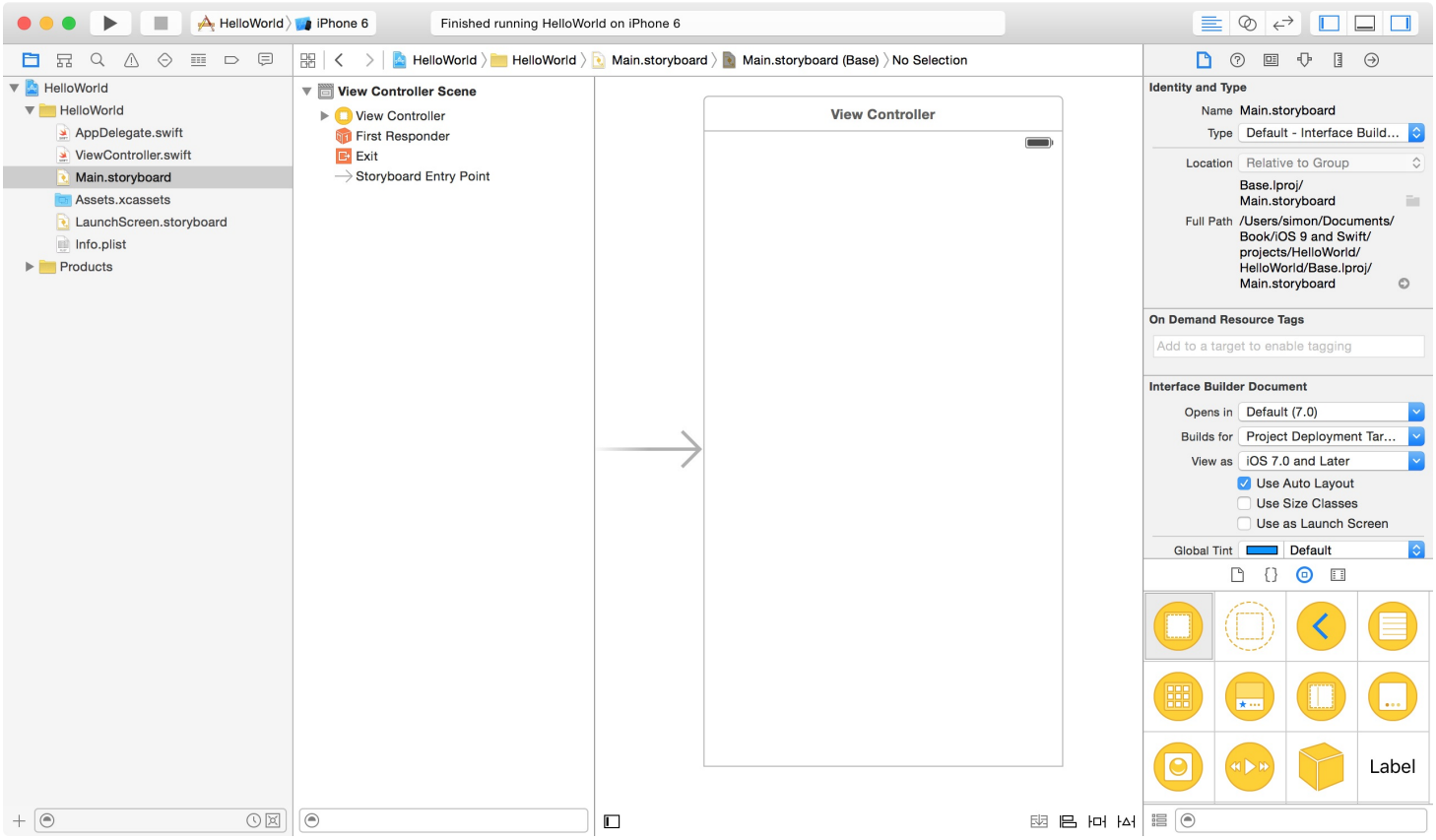

Figure 3-14. View Controller with size classes disabled

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 3-14). 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.

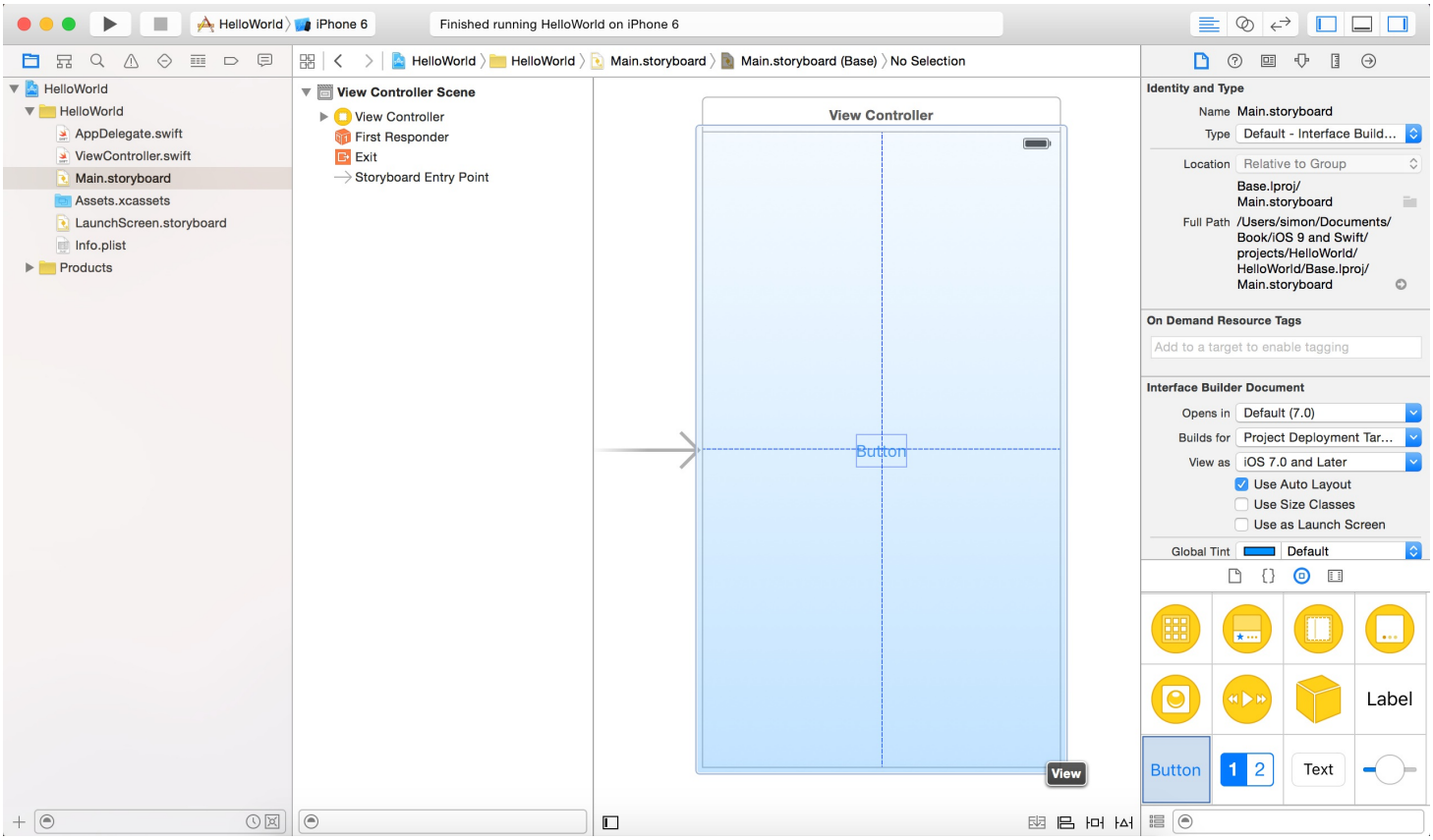

Figure 3-15. Drag the Button to the View

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.

Next, let's rename the button. To edit the label of the button, double-click it and name it "Hello World". After the change, you may need to center the button again.

Figure 3-16. Renaming the button

Great! You're now ready to test your app. Select iPhone 5/5s as the simulator and hit the Run button to run the project, you'll see a Hello World button in the simulator as shown in figure 3-17. Cool, right? However, when you tap the button, it shows nothing. We'll need to add a few lines of code to display the "Hello, World" message.

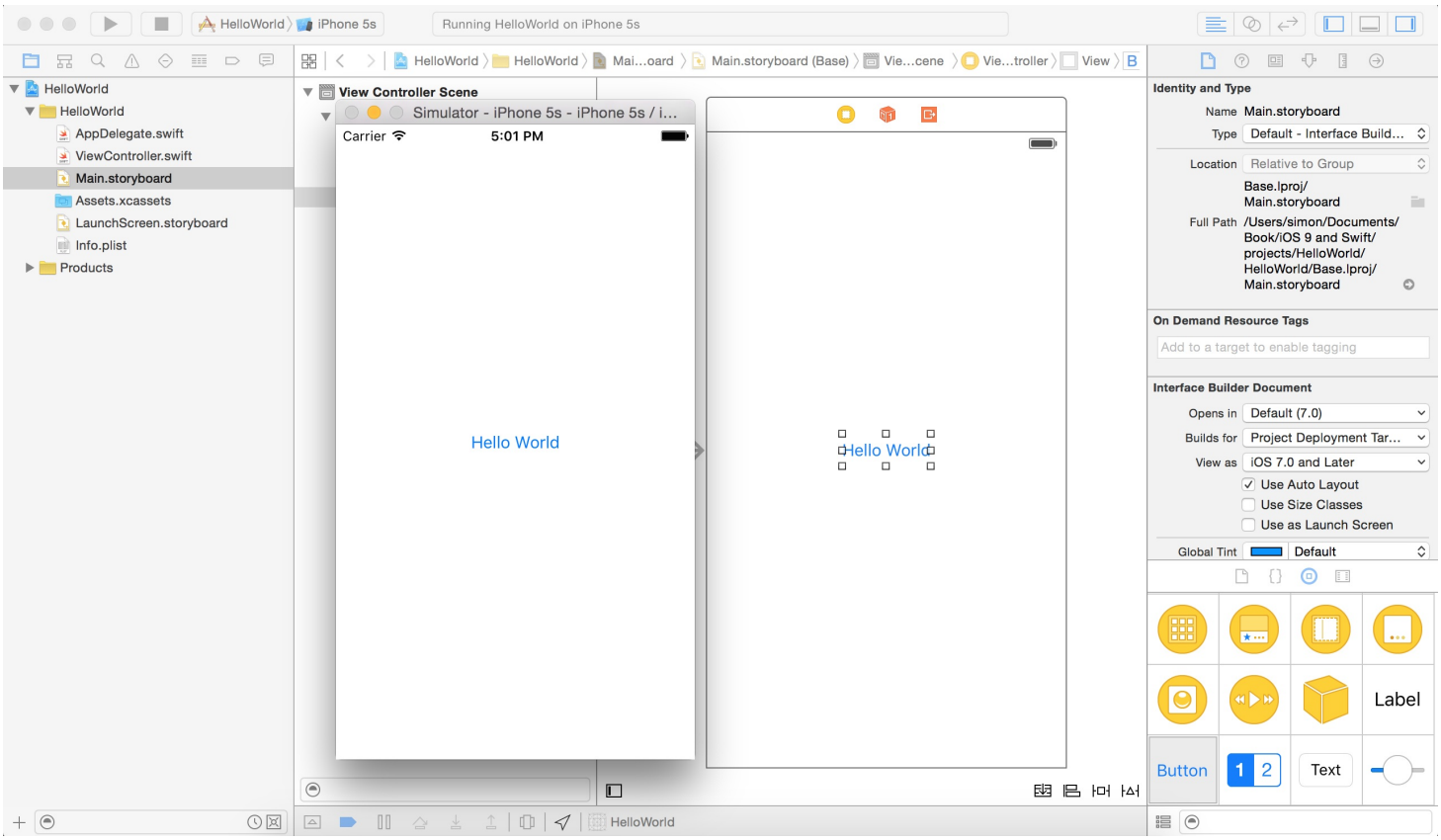

Figure 3-17. Hello World app with a Button

Quick note: 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.

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` file. This file is actually associated with the view controller in the storyboard. 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 `ViewController.swift` file and the editor area immediately displays the source code. Type 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)

}
```

Quick note: I encourage you to type the code, rather than copy & paste it.

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 the 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?

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

Press and hold the control key of the 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.

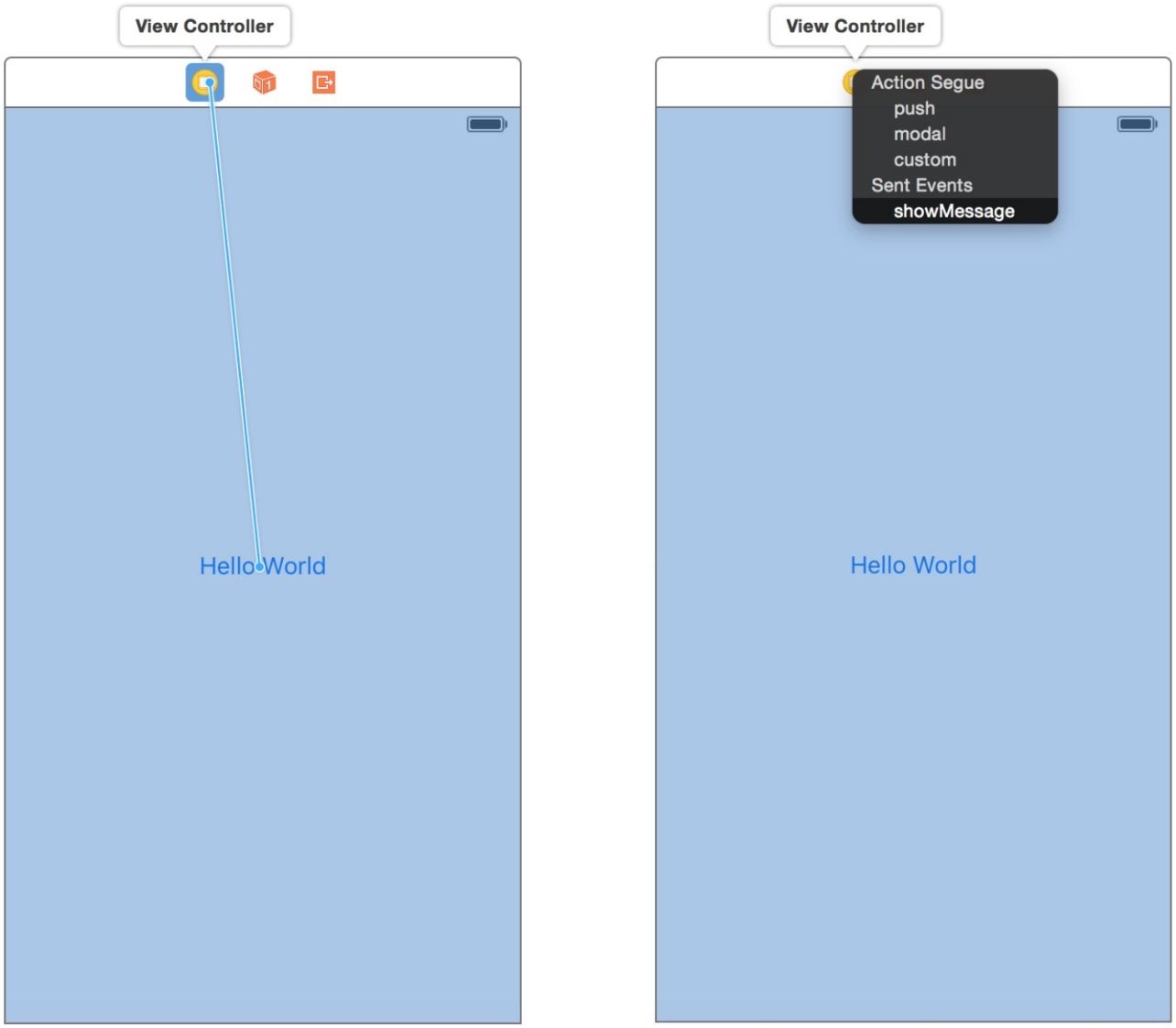

Figure 3-18. Drag to the View Controller icon (left), a pop-over menu appears when releasing the buttons (right)

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.

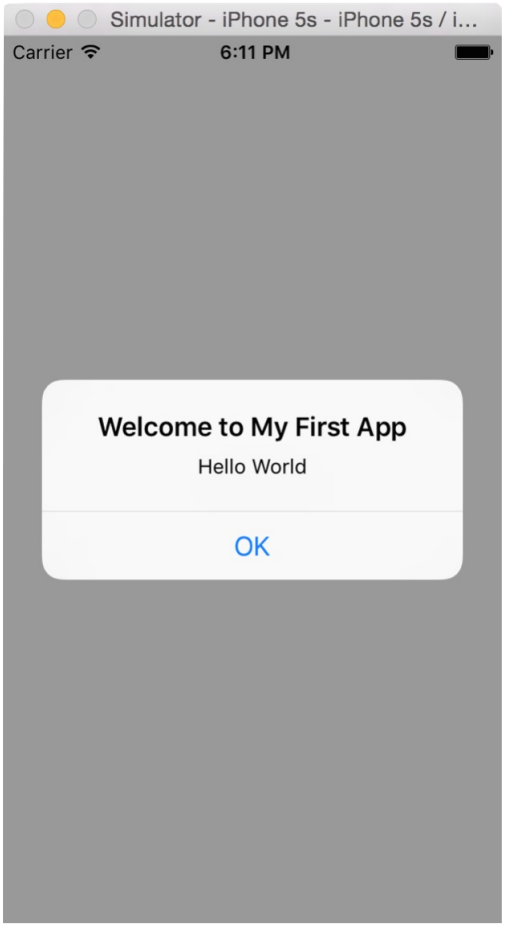

Figure 3-19. Hello World app

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 red or whatever color you want.

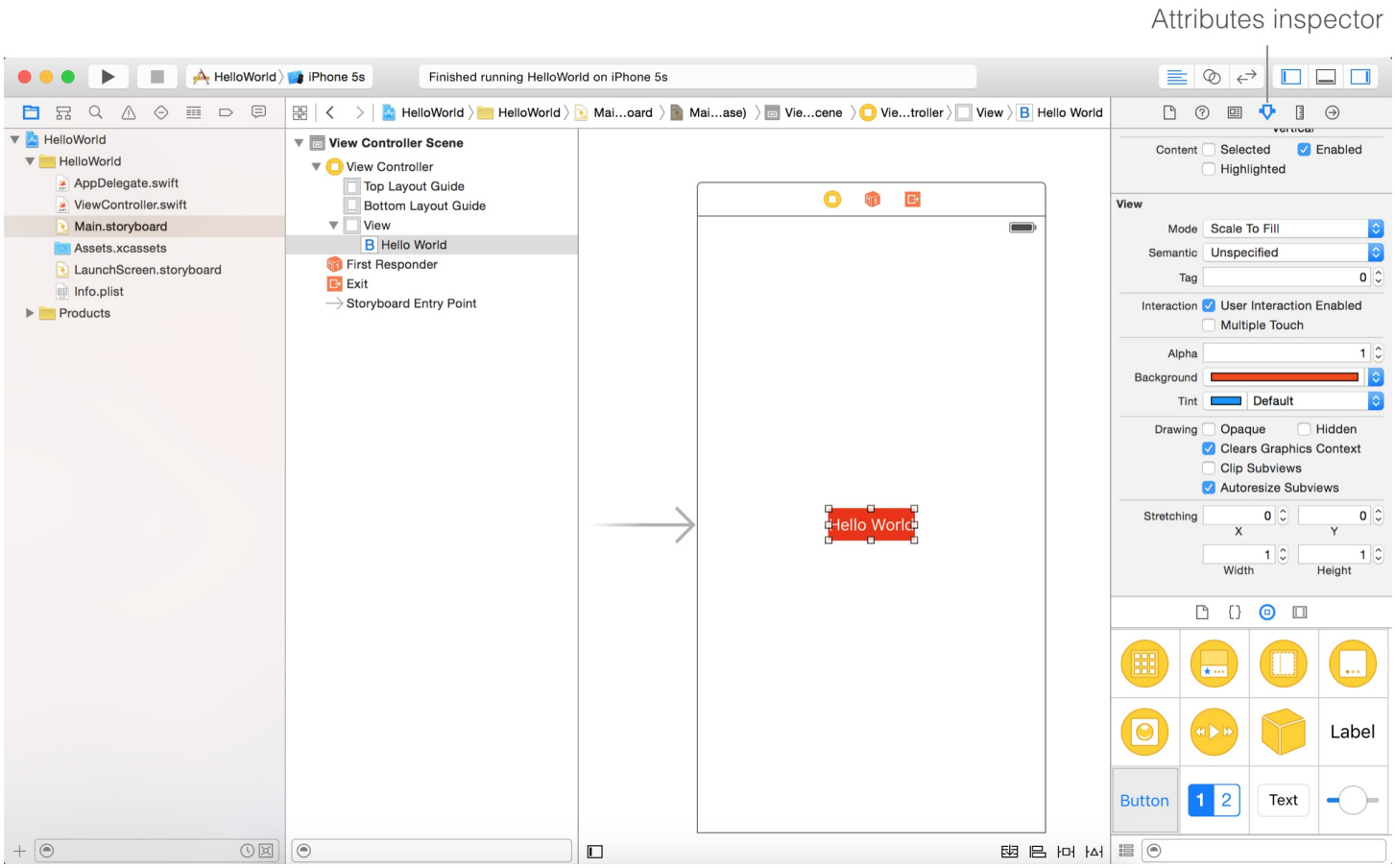

Figure 3-20. Changing the color of the Hello World button

Run the project again and see what you get.

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 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 reference, you can download the complete Xcode project from https://www.dropbox.com/s/m5ayco4iatiba8e/HelloWorld.zip?dl=0.