SECTION I
Hello iOS!

1

Hello iOS!

Hello and welcome to the exciting world of iOS application development. iOS is Apple's operating system for mobile devices; the current version at the time of this writing is 8.0. It was originally developed for the iPhone (simply known as iPhone OS back then), and was subsequently extended and renamed in June 2010 to iOS to support the iPad, iPhone, and iPod Touch.

At its core, iOS is Unix-based and has its foundations in MacOS X, which is Apple's desktop operating system. In fact, both iOS and MacOS X share a common code base. As new versions of mobile operating systems have appeared, Apple has brought over more functionality from MacOS X. This is part of Apple's strategy to bridge the difference between desktop and mobile computing.

With the launch of version 8.0, Apple has not only pushed the boundaries on what is achievable on smart phones and tablet computers, but has also given us a brand new programming language called Swift. This book covers iOS development with Swift only, but at the time of this writing, it is possible to create iOS applications with both the older language Objective-C as well as Swift.

This lesson introduces you to the arena of iOS development.

iOS DEVELOPER ESSENTIALS

Before you get started on your journey to becoming an iOS developer, you will need some essential resources. This section covers these basic requirements.

A Suitable Mac

To develop apps for the iPhone and the iPad using the official set of tools provided by Apple, you will first need an Intel-based Mac running Mac OS X Yosemite (10.10) with a minimum 4GB of RAM and at least 11GB of free space on your hard disk. You do not need a top-spec model to get started. In fact a Mac Mini or a low-end MacBook will work just fine.

Processor speed is not going to make much difference to you as a developer. You will be better off investing your money in more RAM and hard disk space instead. These are things you can never get enough of. A large screen does help, but it is not essential.

A Device for Testing

If you are reading this book, chances are that you have used an iPhone, iPad, or iPod Touch and probably even own one or more of these nifty devices.

As far as development is concerned, there aren't many differences between developing for any of these devices. The obvious differences are screen size and the fact that only iPhones can make phone calls. When you are starting out as an iOS developer, you will test your creations on the iOS Simulator. The iOS Simulator is an application that runs on your Mac and simulates several functions of a real iOS device (more on this later).

At some point, however, you will want to test your apps on a physical device. As good as the iOS Simulator may be, you must test on a physical device before submitting your app to the App Store.

Another good reason to test on a physical device is that the processor on your Mac is much faster than that on the iPhone/iPad. Your app may appear to execute much faster on your Mac (in the iOS Simulator) than it does on the real thing.

If the app you are going to make is targeted at iPhone users, you can also use an iPod Touch as the test device. These are significantly cheaper than iPhones and for the most part offer the same functionality as their phone counterparts.

Most of Apple's devices support iOS 8; however, iOS 8 is not supported for the following:

- iPhones prior to the iPhone 4S
- iPads prior to the iPad 2
- iPod Touch devices prior to the iPod Touch 5th generation

An iOS Developer Account

To develop your apps you will need to download the latest version of Xcode and the iOS SDK (Software Development Kit). To do this, you must sign up to the Apple Developer Program to become a registered developer.

The signup process is free and you can immediately begin to develop your first apps. Limitations exist as to what you can do for free. To submit your apps to the App Store, get access to beta versions of the iOS/SDK, or test your apps on a physical device, you need to become a paying member.

Most of the concepts and apps presented in this book will work just fine with the free membership. The only exceptions would be examples that require the camera, accelerometer, and GPS for which you would need to try the app on a physical device.

You can choose from two forms of paid membership as a registered Apple Developer: Individual and Enterprise.

Individual

The Individual iOS Developer Program costs $99 a year and is for individuals or companies that want to develop apps that will be distributed through the App Store. You can also test/distribute

your apps on up to 100 devices without having to go through the App Store. This form of deployment (without having to submit them to the App Store) is called ad-hoc distribution and is a great way to submit a preview of the app to a client. This form of distribution is covered in detail in Appendix C.

Enterprise

The Enterprise iOS Developer Program costs $299 a year and is for large companies that want to develop apps for internal use and will not distribute these apps through the App Store. With the Enterprise iOS Developer Program there is no restriction to the number of devices on which your in-house application can be installed.

To start the registration process, visit the iOS Dev Center (see Figure 1-1) at `https://developer.apple.com/programs/enroll/`.

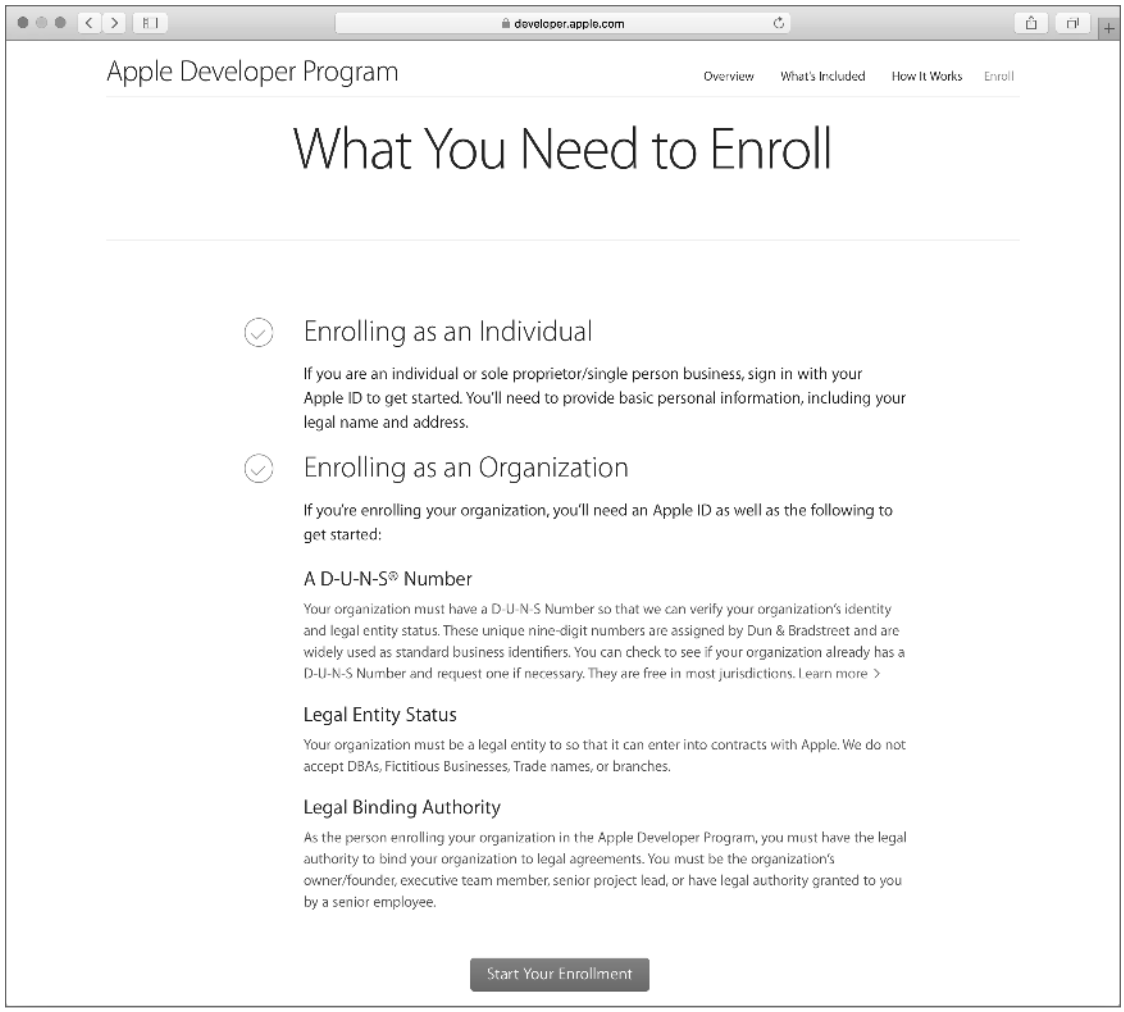

FIGURE 1-1

The Official iOS SDK

The Apple iOS SDK (Software Development Kit) is a collection of tools and documentation that you can use to develop iOS apps. The main tools that make up the SDK are:

- **Xcode:** Apple's integrated development environment (IDE) that enables you to manage your products, type your code, trace and fix bugs (debugging), and lots more.
- **Interface Builder:** A tool fully integrated into the Xcode IDE that enables you to build your application's user interface visually.
- **iOS Simulator:** A software simulator to simulate the functions of an iPhone or an iPad on your Mac.
- **Instruments:** A tool that will help you find memory leaks and optimize the performance of your apps. Instruments are not covered in this book.

In addition to these tools, the iOS SDK also includes extensive documentation, sample code, How-To's, and access to the Apple Developer Forums.

The iOS SDK is available as a free download to registered members (registration is free). However, there are benefits to paid membership, including the ability to debug your code on an iOS device, distribution of your applications, and two technical support incidents a year where Apple engineers will provide you code-level assistance.

Downloading and Installing

You can download and install Xcode 7 for Mac OS X El Capitan and the iOS SDK from the Mac App Store (see Figure 1-2).

If you have a paid membership, you can download the latest version of Xcode as well as prior versions by logging in to the iOS developer portal at `https://developer.apple.com/devcenter/ios/index.action`.

The Typical App Development Process

Whether you intend to develop iOS apps yourself or manage the development of one, you need to be familiar with the basic steps in the development process (see Figure 1-3). This section introduces these steps briefly.

FIGURE 1-2

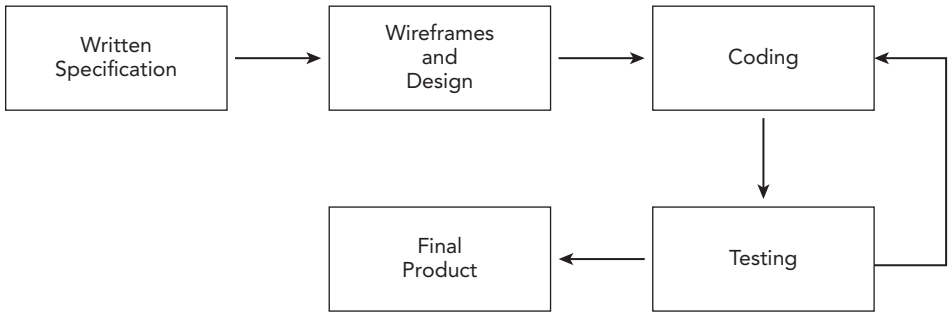

FIGURE 1-3

Writing a Specification

The development of an app begins with a concept. It is good practice to formally put this concept on paper and create a specification. You do not necessarily need to type this specification, although it's a good idea to do so.

At the end of the project you should come back to the specification document to see how the final product that was created compares with the original specification.

As you build your experience developing iOS applications, this difference will become smaller. The specification must address the following points:

- A short description in 200 words or less
- The target audience/demographic of the users
- How will it be distributed (App Store, or direct to a small number of devices)
- A list of similar competing apps
- A list of apps that best illustrate the look-and-feel your app is after
- The pricing model of competing apps and potential pricing for your app

Wireframes and Design

A wireframe is a large drawing that contains mockups of each screen of your app as well as lines connecting different screens that indicate the user's journey through your application.

Wireframes are important because they can help identify flaws in your design early on (before any coding has been done). They can also be used to show potential clients how a particular app is likely to look when it's completed.

There is no right or wrong way to make a wireframe. If it is for your personal use, you can just use a few sheets of paper and a pen. If it is for a client, you might want to consider using an illustration package.

Coding

The actual process of creating an iOS app involves using the Xcode IDE to type your code. iOS apps can be written in either Swift or Objective-C. This book covers iOS development with Swift only.

An iOS app typically consists of several files of Swift code along with resource files (such as images, audio, and video). These individual files are combined together by a process called *compilation* into a single file that is installed onto the target device. This single file is usually referred to as the application binary or a build.

Testing

It might sound obvious, but you must test your app after it has been developed. As a developer, you test your code frequently as you write it. You must also perform a comprehensive test of the entire application as often as possible to ensure things that were working in the past continue to do so.

This form of testing is called *regression testing*. It helps to make a test plan document. Such a document basically lists all the features that you want to test and the steps required to carry out each test. The document should also clearly list which tests failed. The ones that fail will then need to be fixed and the test plan document can provide the replication procedure for the defect in question.

When your app is ready, you will want to list it in the iTunes App Store. To do so involves submitting your app for review to Apple. Apple has several criteria against which it reviews applications and if your app fails one or more of these criteria it will be rejected—in which case you will need to fix the appropriate code and resubmit. It is best to test your apps thoroughly before submitting them in the first place. Distributing your apps via the App Store is covered in Appendix D.

You must always test on a real iOS device before submitting your app for the App Store review process, or giving it to a client to test. Testing on the iOS Simulator alone is not sufficient.

If you are developing for a client, you will probably need to send the client a testable version of your work periodically for review. The recommended way to do this is by using Apple's TestFlight service, which is covered in Appendix C.

Home Screen Icon

Unless you provide an icon for your application, iOS will use a standard gray icon to represent your application in the home screen (see Figure 1-4).

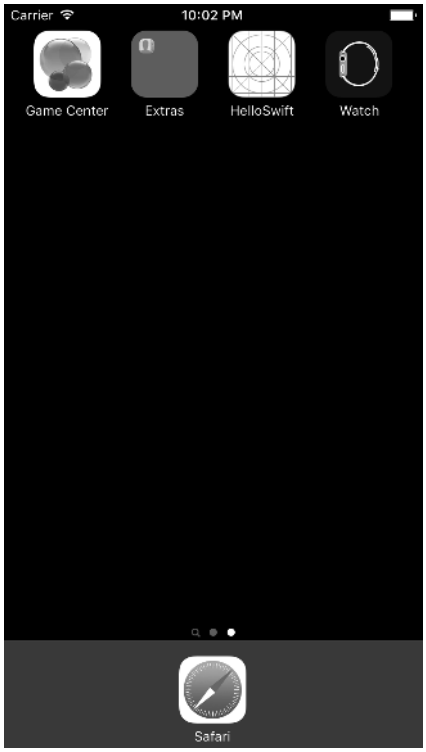

FIGURE 1-4

To replace this icon, you will need to provide one or more PNG files with appropriate dimensions. These dimensions are listed in Table 1-1 and are different for iPhone-based and iPad-based applications.

TABLE 1-1: Home Screen Icon Sizes

DEVICE	ICON SIZE (IN PIXELS)
iPhone 4s	120 x 120
iPhone 5 and iPhone 6	120 x 120
iPhone 6Plus	180 x 180
iPad Retina and iPad Mini Retina	152 x 152
iPad and iPad Mini (without Retina)	76 x 76

You learn to use these icons in this lesson's Try It section.

Application Launch Image

A *launch image* is a placeholder image that you must provide as part of your iOS application. When a user taps your application's icon on the home screen, iOS displays this image while the app starts up.

Once your application has finished loading, iOS gives it control and simultaneously hides the launch image. The overall effect of the launch image is to give your users the perception that your application has launched quickly.

> **NOTE** *The launch image provided as part of your application may not always be used. When an app is suspended into the background state (perhaps because the user tapped the home button on the device), iOS creates a snapshot of the current screen before suspending the app. If the app is resumed within a short period of time then this cached image is used in place of the launch image. However, if the user killed the app, uninstalled it, or hasn't used the app for an extended period of time then the launch image will be used.*

Prior to iOS8, as a developer you had to provide a static PNG version of the launch image for every screen size and orientation that was supported by your app.

While it is still possible to provide static launch images, with the launch of iOS 8 Apple has introduced the concept of a single *launch file*. A launch file is an XIB (or a storyboard file) that describes the user interface for the launch image. An empty document called `LaunchScreen.storyboard` is provided with every iOS project that you create.

The idea behind providing a single launch file over several individual launch images is that iOS will generate the launch images it needs from the launch file for the device on which the app is being used.

You learn to use a launch file in this lesson's Try It section.

TRY IT

In this Try It, you build a simple iPhone application using Xcode 7 that displays the text "Hello Swift" in the center of the screen. You will also provide application icons and a launch file.

Lesson Requirements

- Launch Xcode.
- Create a new project based on the Single View Application template.
- Edit a storyboard in Interface Builder.
- Display the Xcode Utilities area.
- Set up an application icon.
- Set up a launch file.
- Test an app in the iOS Simulator.

> **REFERENCE** *The code for this Try It is available at* `www.wrox.com/go/swiftios`.

Hints

Download and install the latest version of Xcode and the iOS SDK on your Mac; then launch Xcode.

Step-by-Step

- Create a Single View Application in Xcode called HelloSwift.
  1. Launch Xcode.
  2. To create a new project, select the File ➪ New ➪ Project menu item.
  3. Choose the Single View Application (see Figure 1-5) template for iOS and click Next.

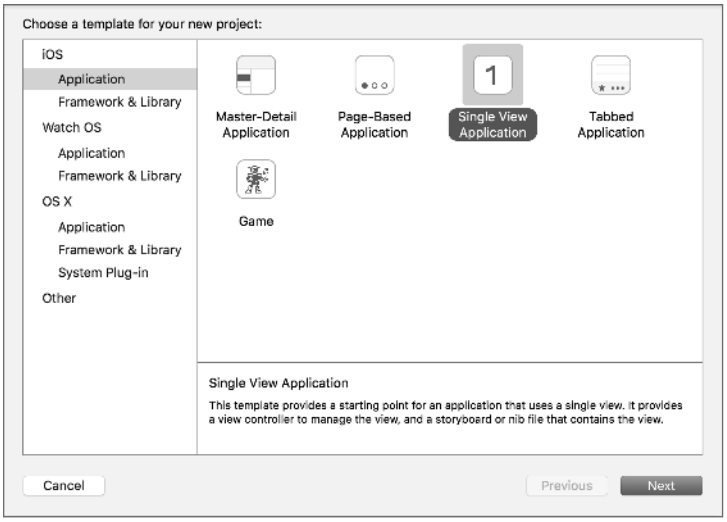

FIGURE 1-5

4. Use the following information in the project options dialog box (see Figure 1-6) and click Next.
   - **Product Name:** HelloSwift
   - **Organization Name:** Your company
   - **Organization Identifier:** com.wileybook
   - **Language:** Swift
   - **Devices:** Universal
   - **Use Core Data:** Unchecked
   - **Include Unit Tests:** Unchecked
   - **Include UI Tests:** Unchecked
5. Select a folder where this project should be created.
6. Ensure the Source Control checkbox is not selected.
7. Click Create.

- Edit the `Main.storyboard` file in Interface Builder (see Figure 1-7).

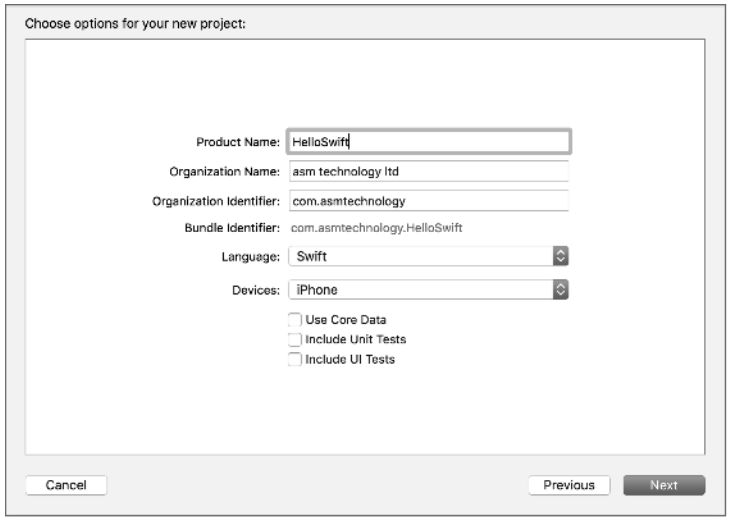

FIGURE 1-6

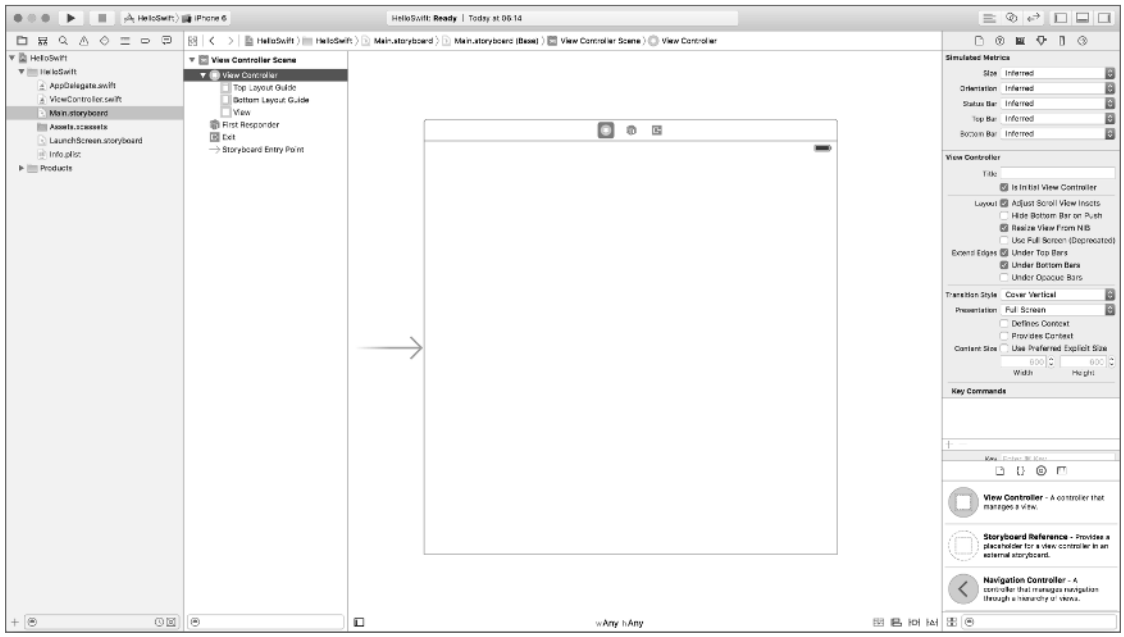

FIGURE 1-7

1. Ensure the project navigator is visible and the HelloSwift project is selected and expanded. To show the project navigator, use the View ➪ Navigators ➪ Show Project Navigator menu item. To expand a project, click the triangle next to the project name in the project navigator.
2. Click the `Main.storyboard` file to select it.
3. Ensure the Attribute inspector is visible by selecting the View ➪ Utilities ➪ Show Utilities menu item.
4. Click the white background area of the default scene in the storyboard.
5. Under the View section of the Attribute inspector, click once on the Background item to change the background color. This is shown in Figure 1-8. Pick any color you want.

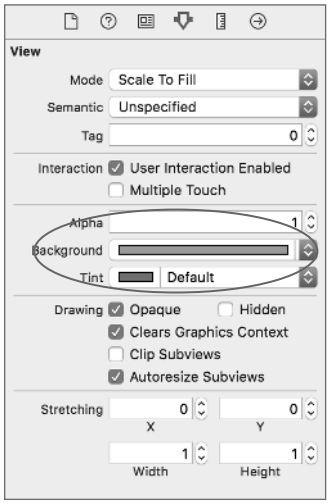

FIGURE 1-8

6. From the Object library in the bottom-right corner, select Label and drop it onto the View (see Figure 1-9). You can use the search box to narrow your choices.
7. Change the text displayed in the Label to "Hello Swift" by editing the value of the Text attribute in the Attribute inspector.
8. Position the label anywhere within the scene using the mouse.

➤ Create layout constraints.

1. Select the label in the storyboard scene by clicking on the label once. Change the size of the label so that the label is large enough to show the text "Hello Swift" fully. To do this use the Editor ➪ Size To Fit Content menu item.
2. Select the label in the storyboard and bring up the Align constraints popup window by clicking the Align button at the bottom right corner of the storyboard (see Figure 1-10).

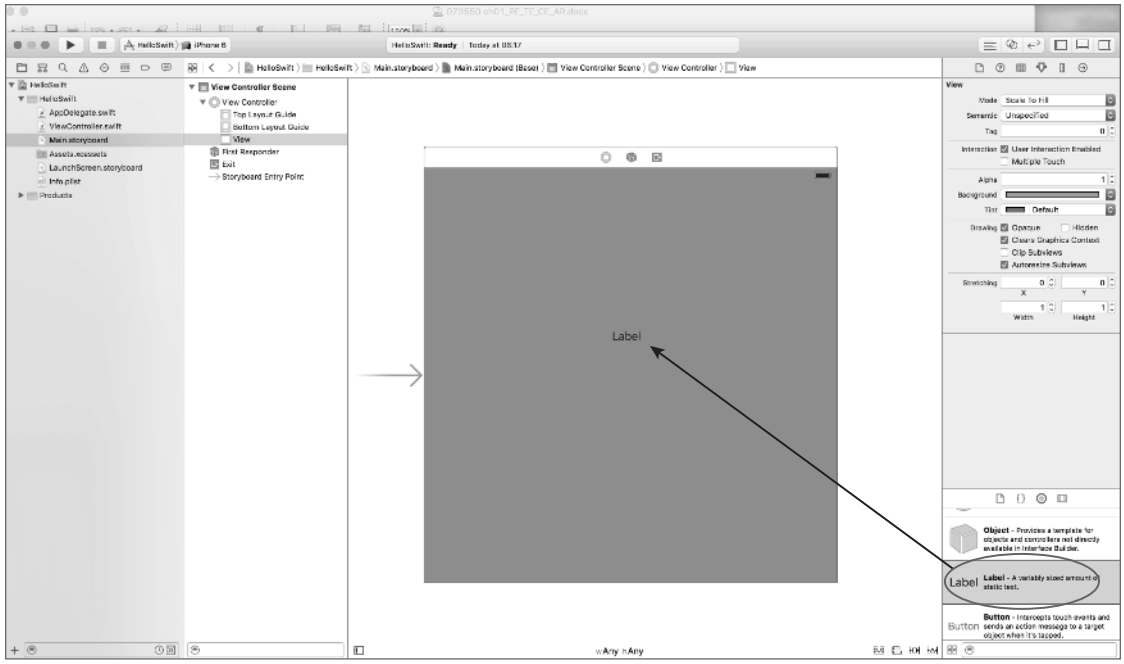

FIGURE 1-9

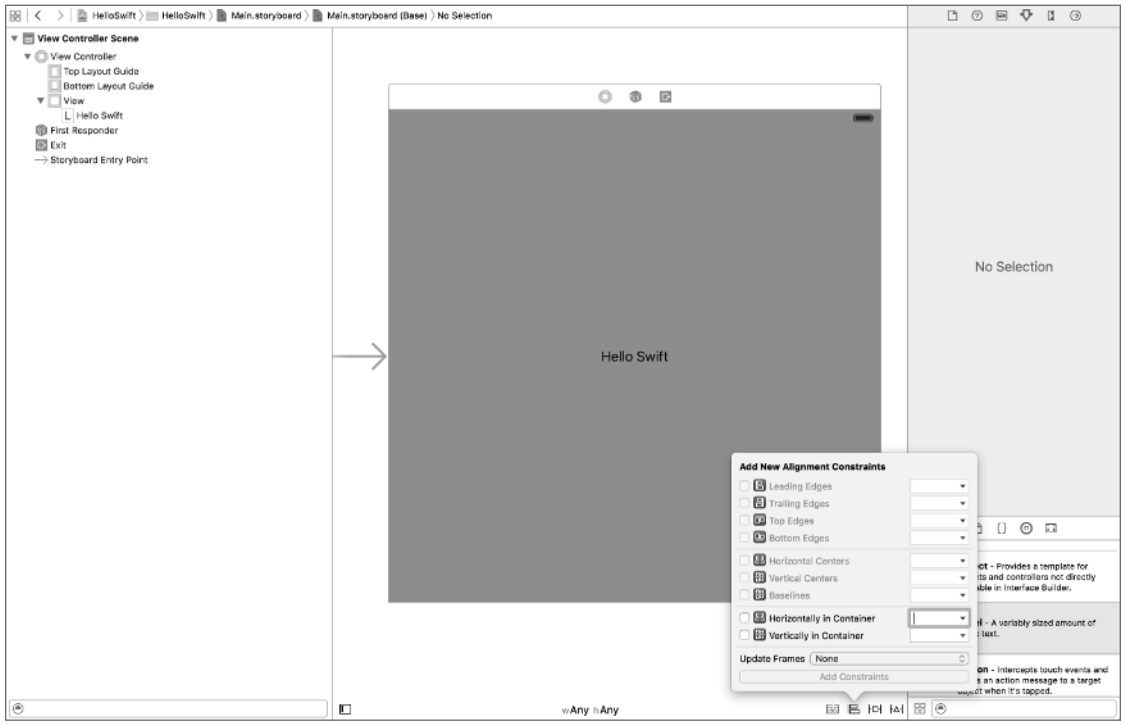

FIGURE 1-10

In this popup window, setup the following options (see Figure 1-11):

- Horizontally in Container: Checked
- Vertically in Container: Checked
- Update Frames: All Frames In Container

FIGURE 1-11

Click the Add 2 constraints button in the popup to apply these layout constraints to the label and dismiss the popup.

> **NOTE** *Selecting All Frames in Container in the Update Frames combo box will force the scene to update the position of the label using the constraints you have just specified.*

- Set up a launch file.
  1. Select the `LaunchScreen.Storyboard` file in the project navigator.
  2. Use the Attribute Inspector to change the background color of the launch file to a different color than that of the scene in the main storyboard.
- Set up an application icon.
  1. Select the `Assets.xcassets` item in the project navigator to open the asset bundle. Select the AppIcon asset within this bundle.
  2. Use drag-and-drop to assign images to the iPhone App and iPad App placeholders. You can obtain the images from the resources available for this lesson on the book's website at `www.wrox.com/go/swiftios`.
     - iPhone App 2x: Use the file `iPhoneAppIcon2x.png`.

- iPhone App 3x: Use the file `iPhoneAppIcon3x.png`.
- iPad App 1x: Use the file `iPadAppIcon1x.png`.
- iPad App 2x: Use the file `iPadAppIcon2x.png`.

After these assignments are made, your scene should resemble Figure 1-12.

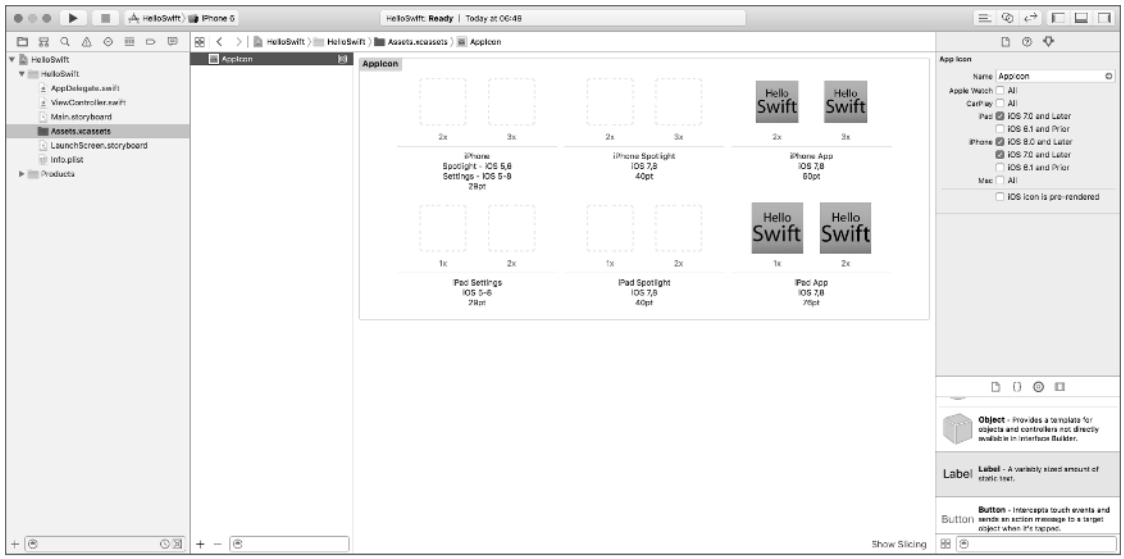

FIGURE 1-12

- Test your app in the iOS Simulator by clicking the Run button in the Xcode toolbar. Alternatively, you can use the Project ➪ Run menu item.

> **REFERENCE** *To see some of the examples from this lesson, watch the Lesson 1 video online at* `www.wrox.com/go/swiftiosvid`.