Covers iOS 7

Get started building your very first iPhone or iPad apps

iPhone and iPad Apps for Absolute Beginners

FOURTH EDITION

Dr. Rory Lewis | Laurence Moroney

Apress®

For your convenience Apress has placed some of the front matter material after the index. Please use the Bookmarks and Contents at a Glance links to access them.

Apress®

Contents at a Glance

Introduction

"Ultimately design defines so much of our experience. I think there is a profound and enduring beauty in simplicity. In clarity. In efficiency. True simplicity is derived from so much more than just the absence of clutter. It is about bringing order to complexity. iOS 7 is a clear representation of these goals. iOS 7 brings with it the most significant set of changes to the user interface since the very first iOS. We see iOS 7 as defining an important new direction, and in many ways—a beginning."

—Jonathan Ive, Designer of iOS

When the original iPhone was released in 2007, it began a wave of innovation in mobility that is far from over. Since then, the user interface has remained largely unchanged—until now, with the release of iOS 7. With this release, Apple has begun the next phase of mobile user interface innovation and has rejuvenated the already vibrant developer market.

If you haven't built for iOS, there has never been a better time to start. This is, as Jonathan Ive mentions above, a beginning.

With this book in your hands, you have everything you need to start on the road to learning the essential skills for iOS 7 development. There are so many concepts that may be confusing to beginners, but when you follow the step-by-step tutorials in this book, you'll begin to knock those concepts down, one by one, until, by the time you end this book, you'll know what it takes to be an iOS 7 developer, and you'll be able to approach online documentation, open source samples, or more complex tutorials confident that you've gotten the foundation down.

This book, like iOS 7 in its way, is a dramatic makeover of the series of "Absolute Beginners" books. It's been rewritten from the ground up to make it more approachable, more intuitive, and more fun than ever before.

I hope you enjoy it, and I look forward to seeing what you can build on iOS 7.

—Laurence Moroney

Chapter 1

Getting Started

If you've picked up this book and have read this far, you're likely interested in being an iPhone and iPad developer. Given the popularity of this platform and the stories of how lucrative development for it can be, you've made a great choice. However, you may also have heard stories of how difficult it can be, how you need to learn abstract or difficult languages and concepts, or how you need to use tools that programmers don't like, or indeed you may have heard how difficult it can be to put your work into the app store, where you can make some money.

Unlearn all that. It's really not that bad, and it can be a fun journey. In this book, you'll go through the basics of what you need to know, from getting the developer tools, to learning how to use them to design and code iOS apps, to debugging and fixing your apps before finally deploying them to the app store.

Your focus will be on the brand new iOS 7 platform. iOS is the operating system (O.S.) that powers these devices. It was first released (although called iPhone OS back then) with the first iPhone back in 2007. In 2013, Apple gave iOS its biggest and most revolutionary update with the release of version 7, called iOS7.

This book is designed for iOS7 development, using a programmer's tool called Xcode and a language called Objective-C. The only thing you'll need is a Mac computer, because Xcode runs only on the Mac, and you'll have to ensure that it's an Intel-based Mac running Mac OS X Snow Leopard or later. Any Mac bought in the last three or four years should be fine. As you'll see in the "Getting Xcode" section later in this chapter, you get Xcode from the app store, so you'll need an account there too. It's free to get an account, and Xcode is free to download.

You don't *need* an iPhone or iPad in order to develop for them, as Xcode gives you a simulator for both of these devices. However, if you want to test on a physical device, which sometimes is necessary, you'll need a developer account for iOS development, at a cost of $99 per year.

In this chapter you'll get started by downloading and installing Xcode and taking a look at how it works. You'll also see how to sign up for a developer account and how to access it.

Getting a Mac

First off, you will definitely need a Mac to work with. It doesn't have to be the latest, most expensive model, but as mentioned in the previous section, it must be an Intel-based Mac running Mac OS X Snow Leopard or later. We wrote this book using two Macs—a 2011 model iMac and a 2011 model MacBook Air.

You can see the different options available to you at `http://www.apple.com/mac/`, displayed across the top of the screen, as shown in Figure 1-1.

Figure 1-1. *Buying a Mac*

At the time of writing, the latest version of OS X, called Mavericks, was in preview, as indicated in Figure 1-1. Of course, you don't *need* Mavericks, but if you have it, you're fine to continue.

When buying a Mac, there are several important attributes that you should consider:

- **Processor Speed:** The faster the processor, the quicker your programs will load, the quicker they'll update the screen, and the quicker Xcode will compile your apps
- **Memory:** The more memory the better. OS X will use memory as it runs applications. If it needs to use more memory than is available, it has to temporarily store information on disk. Accessing the disk is slower than accessing memory, so the more memory, the faster your machine will generally operate
- **Disk Space:** As you download or create information, you'll take up disk space. The more space you have, the more time you have before you run out!
- **Disk Type:** Most machines have a hard disk, but some have a flash drive, also known as a solid state drive (SSD). Hard disks are cheaper, so machines with flash drives tend to be more expensive, or, because of the expense, will have less capacity. The big advantage of flash drives, however, is that they are very fast—in some cases, almost as fast as memory. Thus, a machine with a flash drive can be very responsive, but may have less available storage. MacBook Airs use flash drives, which are generally optional on the other machines.
- **Screen Resolution:** Screen resolution is the number of dots on your screen in width and height. So, for example, you might see 1366x768 as the resolution on a machine, indicating that the screen has 1366 dots (pixels) across and 768 down. The bigger the resolution, the more you'll see. When developing, it's good to have a big screen resolution, giving you plenty of room to view Xcode, the simulator, and a browser for the documentation all at the same time.

We like to use an iMac for our main development, because it comes with a large monitor that offers very high resolution. We use a MacBook Air as a backup, because it's light and very fast (due to its flash drive).

Getting Xcode

OSX Snow Leopard and later include an app store application through which you can get free software or purchase paid software. Apple distributes Xcode using the app store. If you're not familiar with the app store, you can see its icon in Figure 1-2. It's the one in the center with the pencils and ruler arranged like the letter "A."

***Figure 1-2.** The Mac app store*

When you launch the app store, you'll see a home screen with the latest "Editor's Choice" applications, as well as others that are new and noteworthy. At the top right side of the screen you'll see a search box. Type "Xcode" into this box and press the "Enter" key. See Figure 1-3.

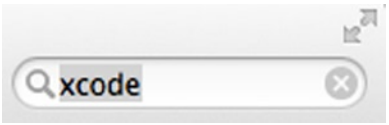

Figure 1-3. *Searching for Xcode*

The app store will return a bunch of apps that match this search term. It should look something like Figure 1-3, but don't worry if it's a little different—apps are being added all the time. Just make sure that you can see Xcode itself, with its "hammer" icon, as shown in Figure 1-4.

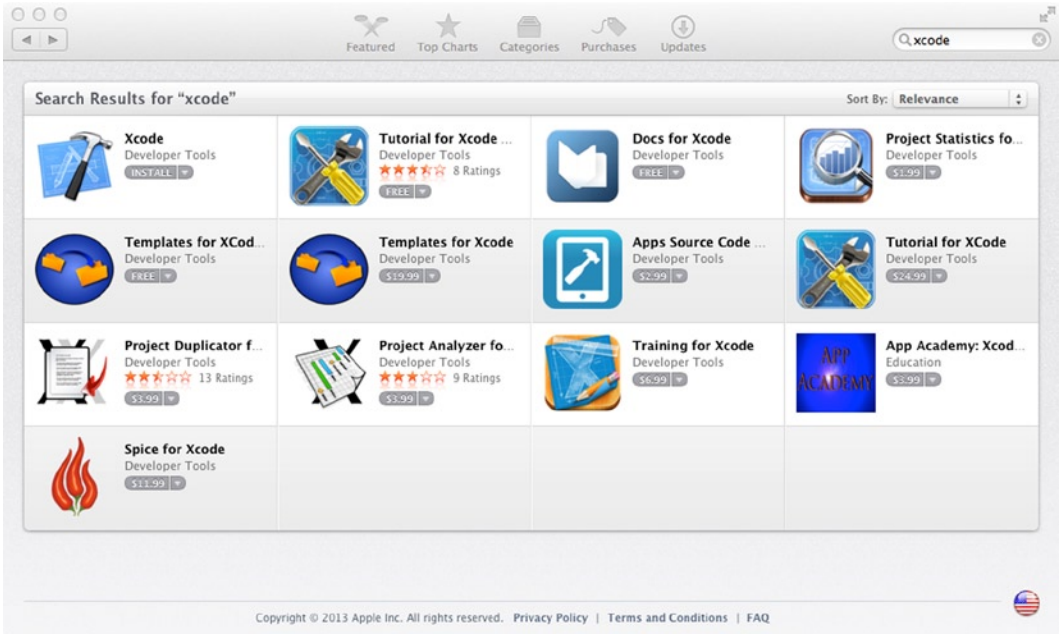

Figure 1-4. *Searching for Xcode*

If you have a store account, you can go ahead and get Xcode now. If not, read the next section, where you'll see how to get an account that you can use to acquire and download Xcode.

Getting a Store Account

With the app store open, select the "Store" menu at the top of the screen, and from this select "Create Account…." See Figure 1-5.

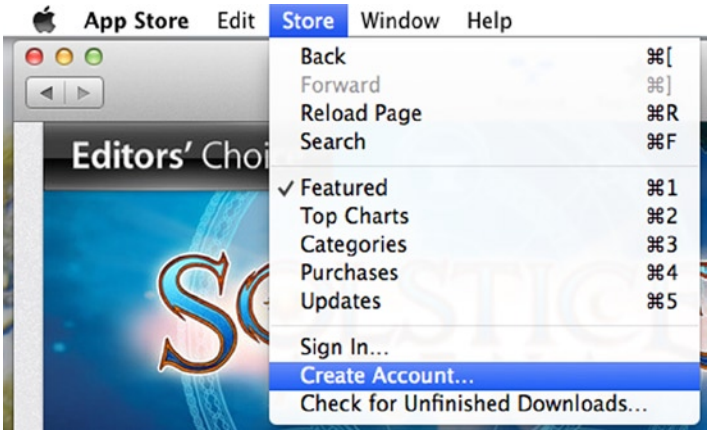

Figure 1-5. *Getting a store account*

You'll be taken to the "Welcome to the App Store" screen. From here, you can click "Continue." You can see it in Figure 1-6.

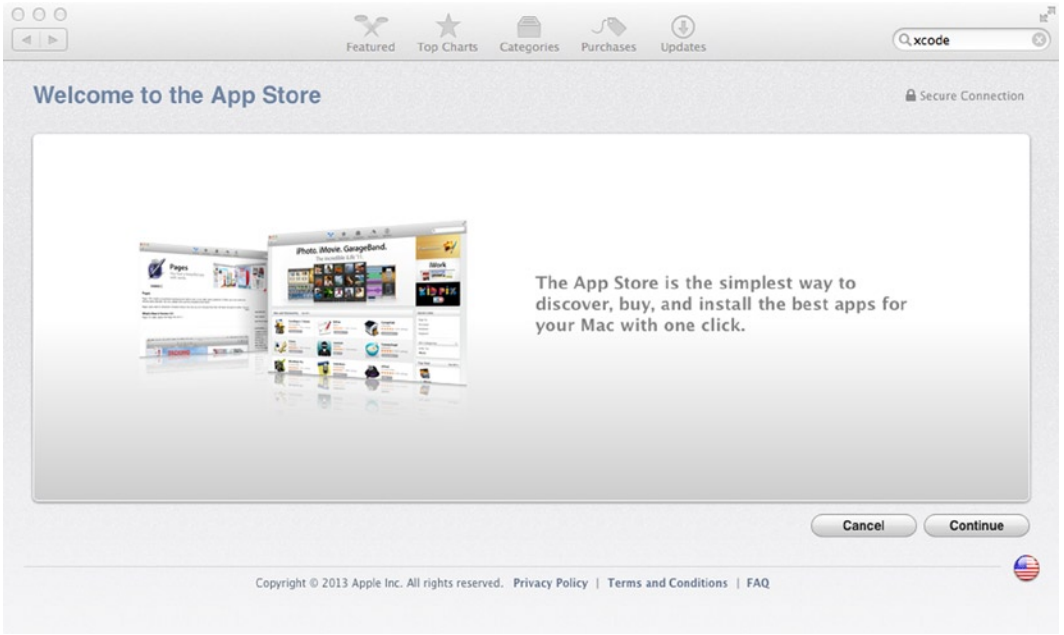

Figure 1-6. *Step 1 of setting up an app store account*

The next step gives you the terms and conditions of the store, as well as Apple's privacy policy. If you agree to the terms, check the box and then click the "Agree" button. See Figure 1-7.

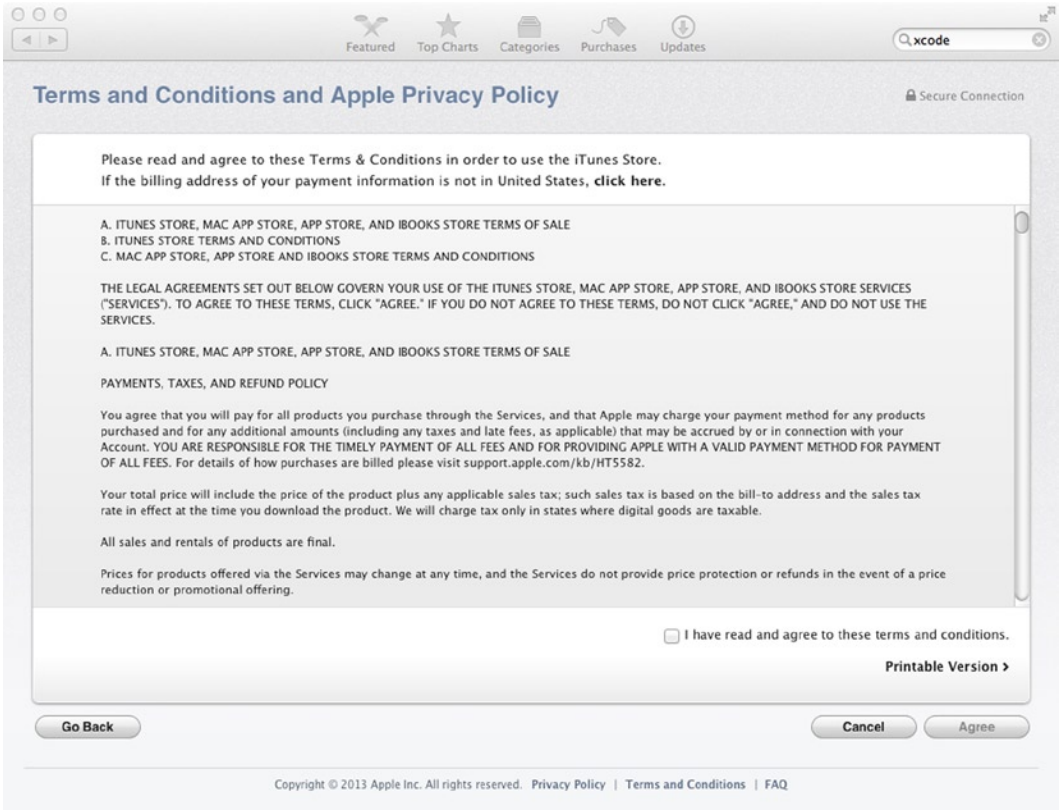

Figure 1-7. *Step 2 of the app store signup*

On the following screen, you will set up the details of your Apple ID. You need to provide an email address, security information, a backup email address, and your date of birth. Fill out values for the security info that you'll remember (and not silly answers like ours), and when you're done, click "Continue." You can see all this in Figure 1-8.

***Figure 1-8.** Setting up your Apple details*

When you're done with this, you'll reach the final screen, where you will set up your payment method. Fill out the details as shown and click "Create Apple ID." Your Apple ID will be created, and Apple will send an email to the address of the ID. You'll have to click on a link in this email to confirm your address, and then you'll be ready to use your new Apple ID.

Downloading and Installing Xcode

When you search for Xcode in the app store, you'll see it with a link that allows you to install it (if you've downloaded it before), or notifies you that it's free, as in Figure 1-9.

Figure 1-9. Accessing Xcode in the app store

Click on the button, and it will change to a green “Install App” button. Click on this, and you’ll be asked to log in with your Apple ID.

Figure 1-10. Signing into the store

Use your Apple ID and password. Xcode will begin to download. You’ll see its progress from within LaunchPad. See Figure 1-11.

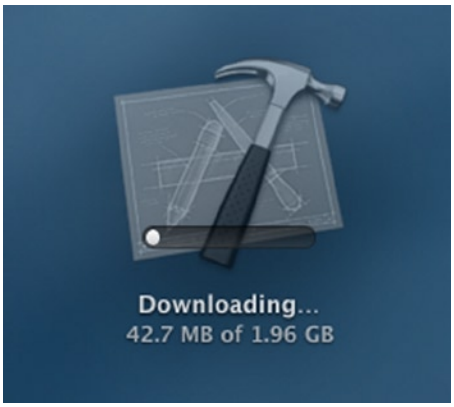

Figure 1-11. Downloading Xcode

It’s a big download, so it will take a bit of time. While you’re waiting, you could check out the page for Xcode in the app store. You can get this by clicking on the word “Xcode” in the search results. Its page looks like Figure 1-12.

Figure 1-12. *Xcode's app store page*

Here you can learn about the features of Xcode, as well as read reviews that other users have posted about it. Previous versions were a little confusing, as the app store only downloaded the installer, and then you had to install explicitly from the download, which wasn't documented very well. This led to a lot of negative reviews.

However, version 5, which you should be installing here, doesn't require this—Xcode will download and install and be ready to use right away!

Running Xcode

You can launch Xcode from launcher. When you do so, and it runs for the first time, you'll see the "Welcome to Xcode" screen. See Figure 1-13.

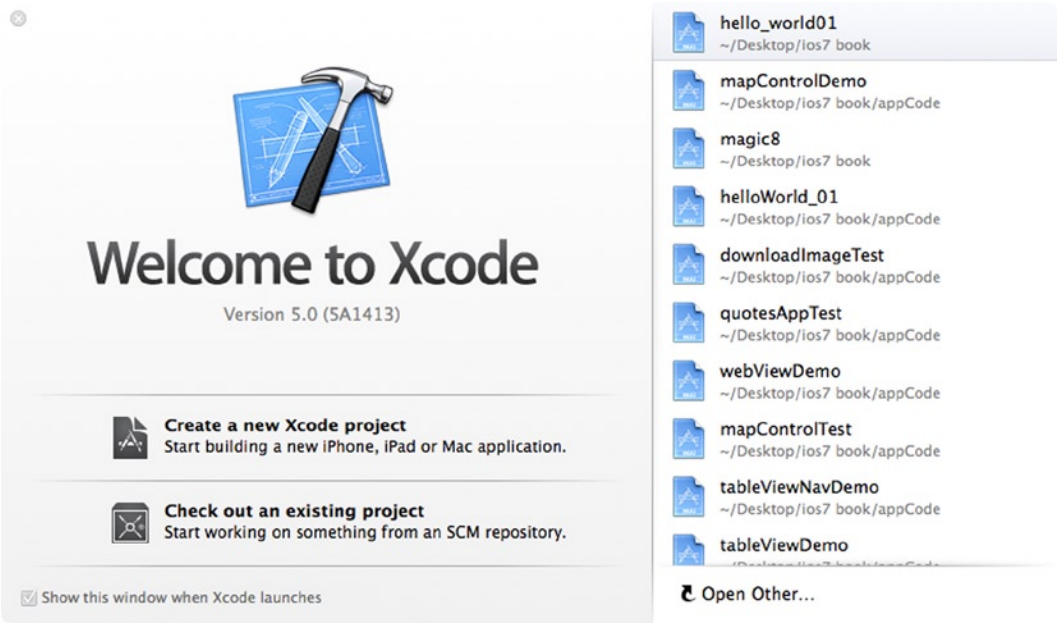

***Figure 1-13.** Xcode's welcome screen*

This screen allows you to create a new Xcode project, and from there to choose a number of different project types. You'll learn about these starting in Chapter 2, where you'll create your first application. Before doing that, it's good to switch gears for a moment and take a look at Apple's developer portal, which is a great source of information and resources for you as you develop applications.

The Apple Developer Portal

Apple's developer portal is at `https://developer.apple.com`. It has sections for iOS7, OS X Mavericks, and Xcode 5. See Figure 1-14.

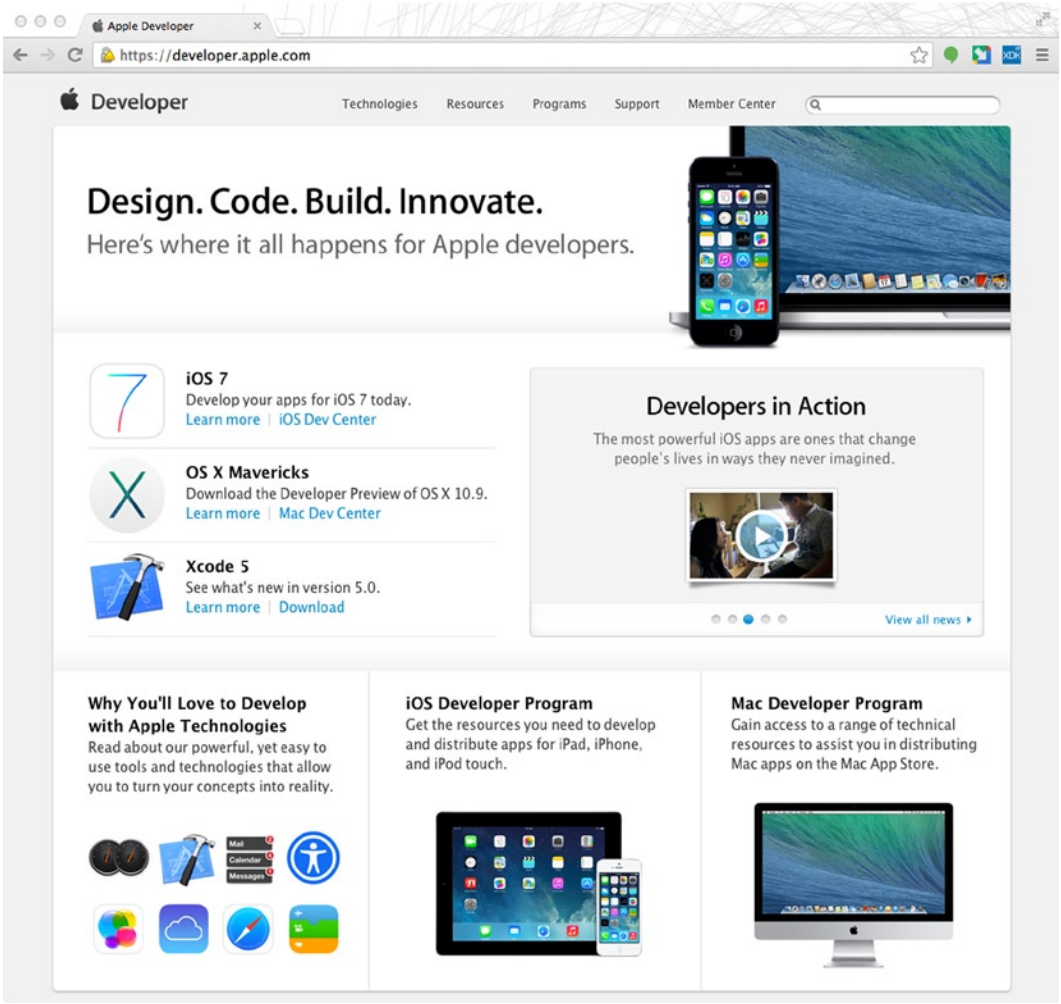

Figure 1-14. *The Apple developer portal*

Click on the iOS dev center link, and you'll be taken to the iOS dev center page, which contains documentation, videos, and more. Most of the information is freely available—you don't need a developer account. If you want to test on physical devices or deploy to the app store, you'll need to register as a developer. This costs $99 per year. You'll step through how to do that in Chapter 3.

The documentation and videos section (see Figure 1-15) is particularly useful, giving you "Getting Started" guides as well as sample code and other references. If you are an absolute beginner, once you've gone through this book and gained some confidence in building iOS apps, go back and visit that section—it's really good, but does require you to have a little knowledge.

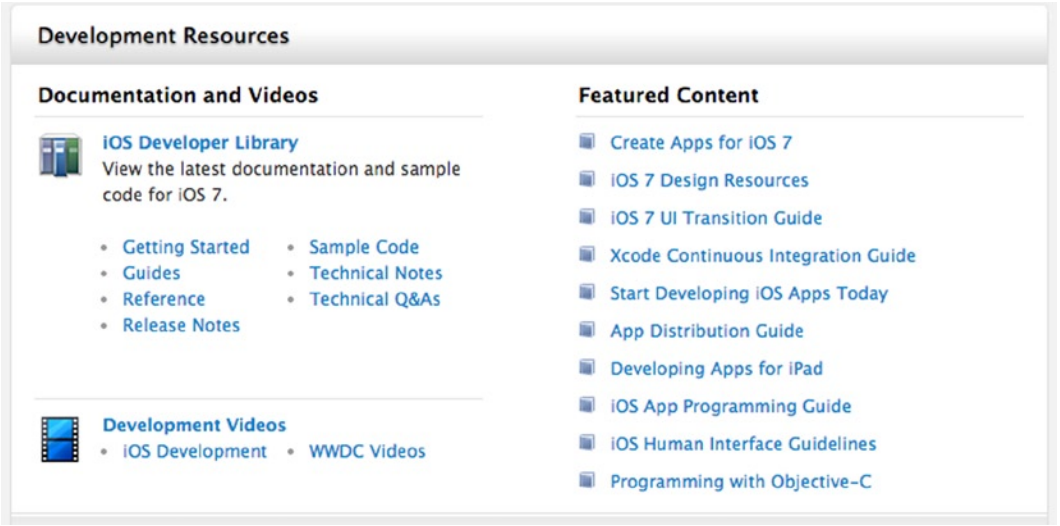

Figure 1-15. *Development resources*

At the very least, consider looking at the "Getting Started" section once you've worked through the tutorials in this book; you'll find it very rewarding!

Summary

This chapter was your introduction to development for iOS 7 using Xcode. From it you learned where the tools are available and how to download and install them. Starting in the next chapter, you'll use these tools to build your first app, and in Chapter 3 you'll learn how to deploy it to your hardware using a developer account.

Chapter 2

Your First iOS Apps

The first app you'll build is a basic "Hello, World" app where, despite the simplicity of the app, you'll implement some of the functionality that is found in more advanced applications, including taking user input and having your code update the user interface as a result. By the time you finish this chapter, you'll know how to run the app in the following three ways:

iPhone app on iPhone simulator: This shows you how to run an iPhone app in the iPhone aspect ratio on an iPhone-sized emulator.

iPhone app on iPad simulator: iPhone applications can be run on iPads too, with a special "2x" mode that enlarges the user interface to make it fit the screen. You'll see how to test your iPhone app with the iPad simulator to see how it would look in this scenario.

iPad app on iPad simulator: Applications can also be written specifically for the iPad, taking advantage of the larger screen resolutions. You'll see how to do that too!

Note Supplementing this book are a number of screencasts of the examples in this book. If you prefer video instead of step-by-step reading, we hope they'll be a great resource for you. You can access these, as well as blogs and help forums around the book, by visiting `ios7developer.com`.

Creating Your First iPhone App

In this first example, you're going to create an app that gives you a button and a label. When you tap the button, text will appear on the label that says "Hello World!" This is a *very* simple app, but you're going to spend quite a bit of time, and quite a few pages, on it. That is because an app like this, despite its simplicity, implements many of the concepts found in larger, more sophisticated apps, making it a really useful learning tool.

Use Xcode to create your app. Launch Xcode, and the “Welcome to Xcode” screen will appear. See Figure 2-1.

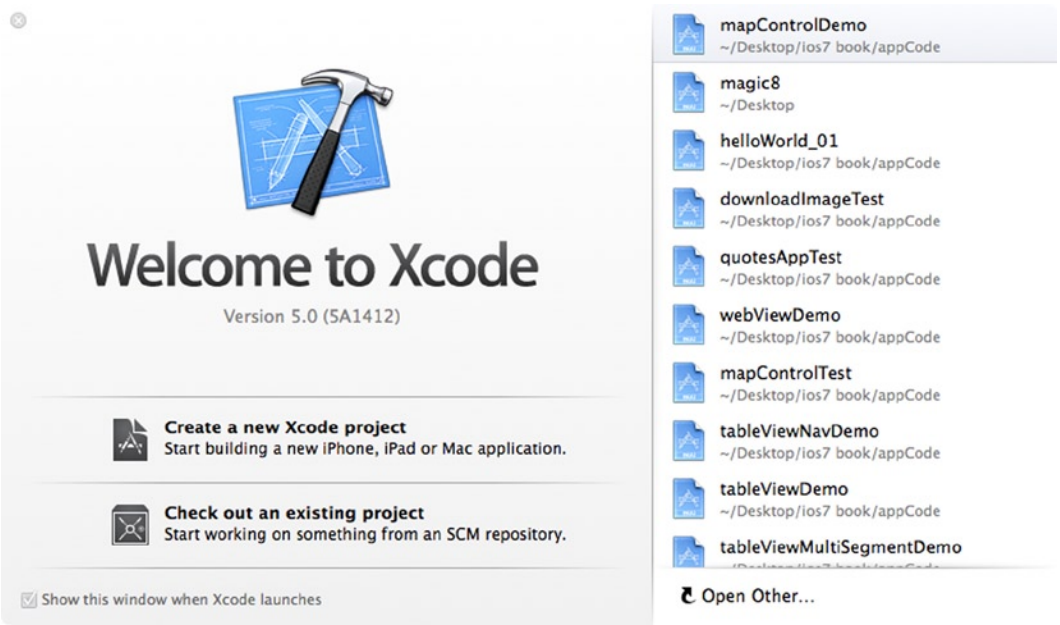

Figure 2-1. *The “Welcome to Xcode” screen*

Click the “Create a new Xcode project” option, and you’ll see a list of templates for your new application.

Your first application will use the “Single View Application” template. Select it as shown in Figure 2-2, and click “Next.”

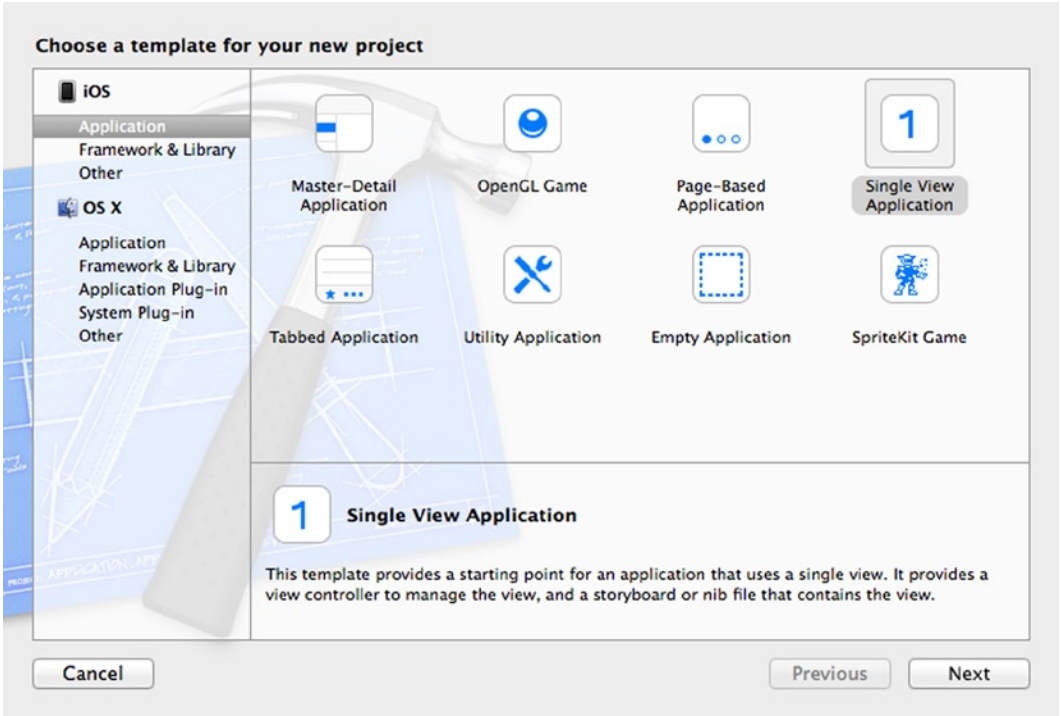

Figure 2-2. *Selecting your application template*

When you click "Next," you'll see a new dialog that requests the options for your new project.

These settings are shown in Figure 2-3, and the recommended settings are:

Product Name: This is the name of your app. For now use "helloWorld_01."

Organization Name: This is the name of your organization, or yourself if you are an individual developer.

Company Identifier: This is the identifier for your company. It is written in reverse-domain-name format. So, for example, if your company is `ios7developer.com`, you would use "com.ios7developer" here. The reason for this is that once you deploy your app to the app store, you could have different apps with the same name, written by different people. Our "Hello, World" will be `com.ios7developer.helloWorld_01`, whereas yours will be different.

Class Prefix: This is used to organize your classes into namespaces, which defaults to matching your company identifier. For the sake of simplicity, erase the class prefix. It will make your initial code easier to read.

Devices: You'll see the drop-down list giving you the options "iPad", "iPhone," and "Universal." This defaults to "iPhone," so keep it that way. A "Universal" selection is one that runs on both iPhone and iPad.

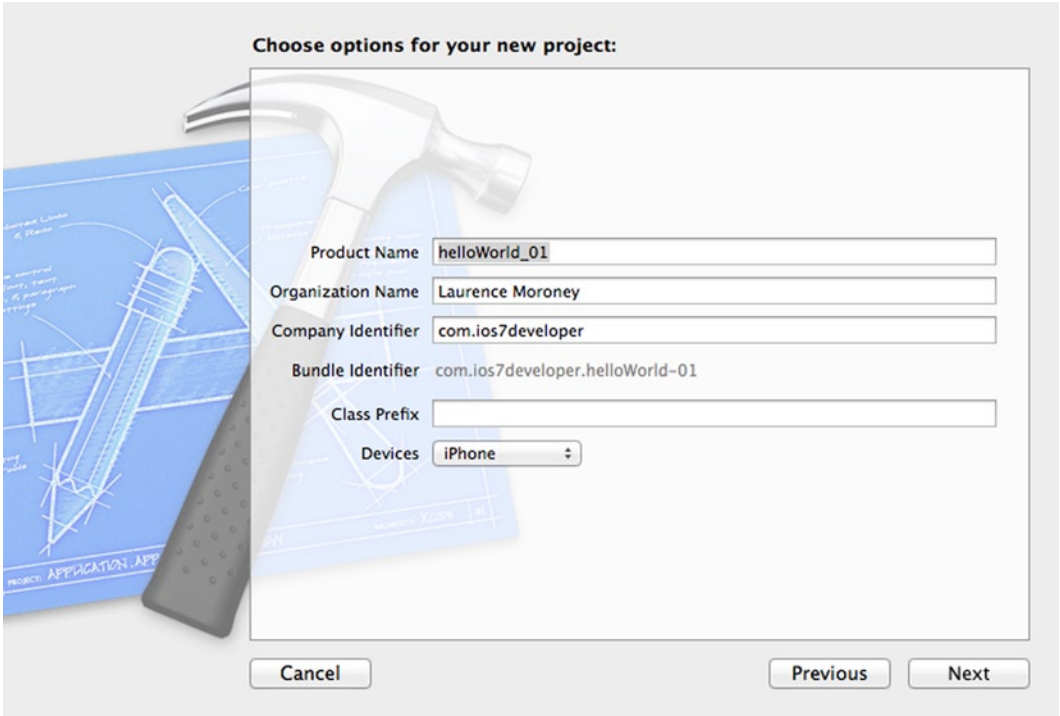

Figure 2-3. *Setting the options for your new project*

Click “Next,” and you’ll get a dialog that asks you where you want to create your code. It will likely default to your desktop. If not, use the drop-down list at the top of the screen to specify your desktop and click “Create.” You can see this in Figure 2-4.

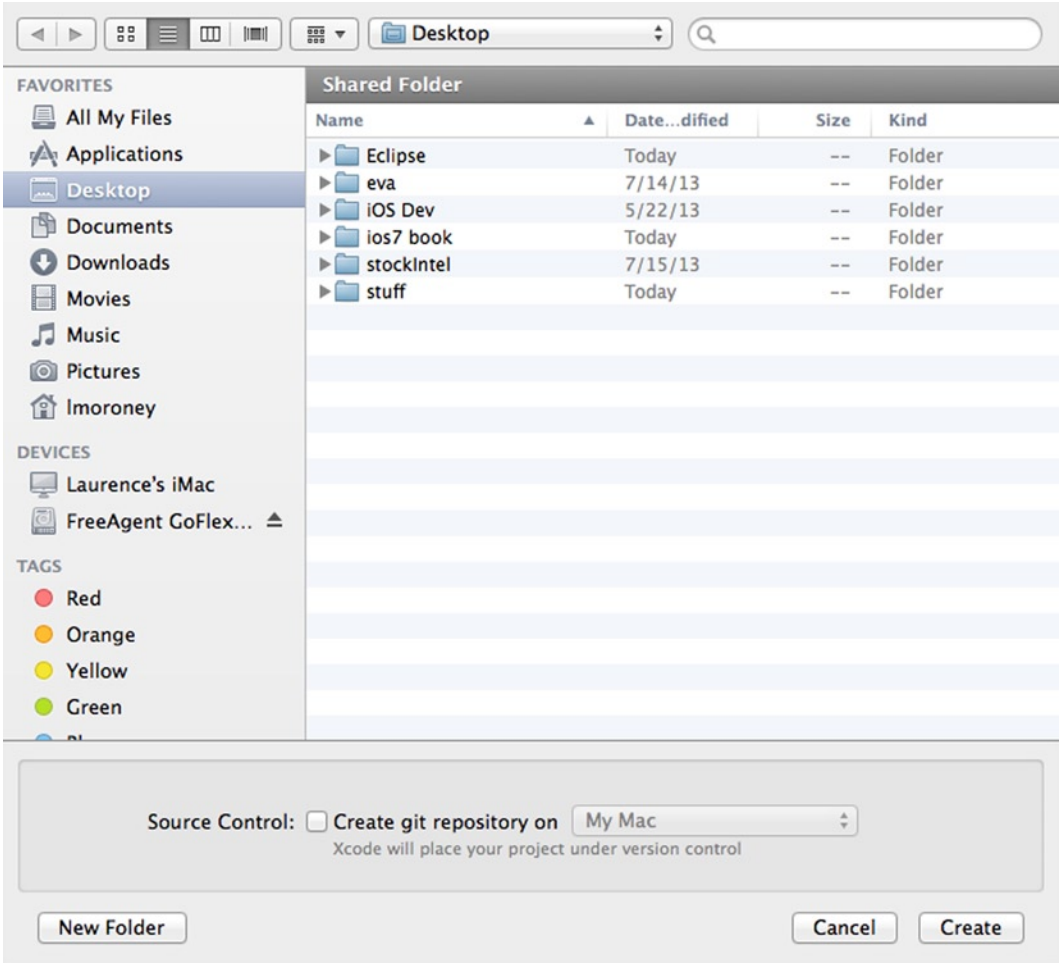

Figure 2-4. *Choosing your source location*

Note that you should keep the "Source Control" checkbox at the bottom unchecked. If you were to check this box, you would be enabling Source Code control through git. This allows you to work with other developers in such a way that you can check changes in and out, avoiding multiple people touching code at the same time. It's beyond the scope of this book, but if you're interested in learning more about git, the 'Pro Git' book by Scott Chacon is available for free here: `http://git-scm.com/book`.

Once you click "Create," Xcode will create a folder on your desktop containing all the source code and metadata that your application needs. Xcode will also launch the workbench that you'll use for editing your user interface and code. You'll see that in the next section. Congratulations—you've created your first iPhone app project!

Xcode and Your Project Files

In the previous section you created your first app, called `helloWorld_01`, and saved it on the desktop. Xcode then launched, showing your application.

Figure 2-5 shows what the top-left-hand side of Xcode will look like. If you don't see the code files listed, make sure the folder icon in the toolbar is selected, as shown in Figure 2-5.

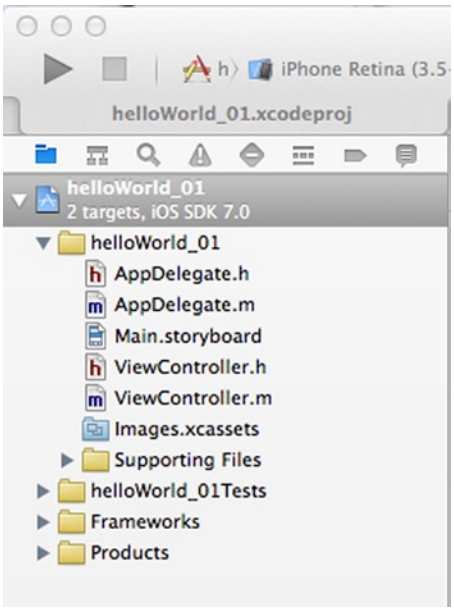

Figure 2-5. *Exploring your Xcode project*

There are lots of files shown here, and you'll learn what each does as you work along through this book. You can see that there are two pairs of files (`AppDelegate` and `ViewController`), each with a ".h" and a ".m" file. These are called *classes,* and they have the code for your application in Objective-C. `AppDelegate` has the code for the shell of your application, and `ViewController` has the code for the default view of your application. Think about a single-view application as a shell containing a view. The shell handles all the application-level stuff, such as launching, shutting down, and so forth, and each "screen" in your app has a view. The code for this view is called a "controller." As this app only has one screen, it has one controller piece of code, and that's the `ViewController.h` and `ViewController.m` files. These terms come from a common pattern used in software development called "Model-View-Controller," or just MVC. Under this pattern, applications can be built by creating a *Model* of your data, a *View* that the user has on your data, and a *Controller* that manages the interaction between the two.

The "Supporting Files" folder contains other files that your app will need, including the ".plist" file that contains metadata about your application and the `main.m` file that is the starting point of your application as it runs.

Your user interface is the `Main.storyboard` file. You'll edit that in the next section.

Don't worry if some of this seems overwhelming right now—there are a lot of concepts to take in, and you'll go through them step by step. Before you know it, it will become second nature!

Using Interface Builder to Create Your "Hello, World" App

Using Xcode, select `Main.storyboard`, and the storyboard editor will open. On the left-hand side of the screen, you'll see a menu drop down that reads "View Controller Scene."

Drop this down to view all the options underneath it and select the "View Controller" entry, as shown in Figure 2-6.

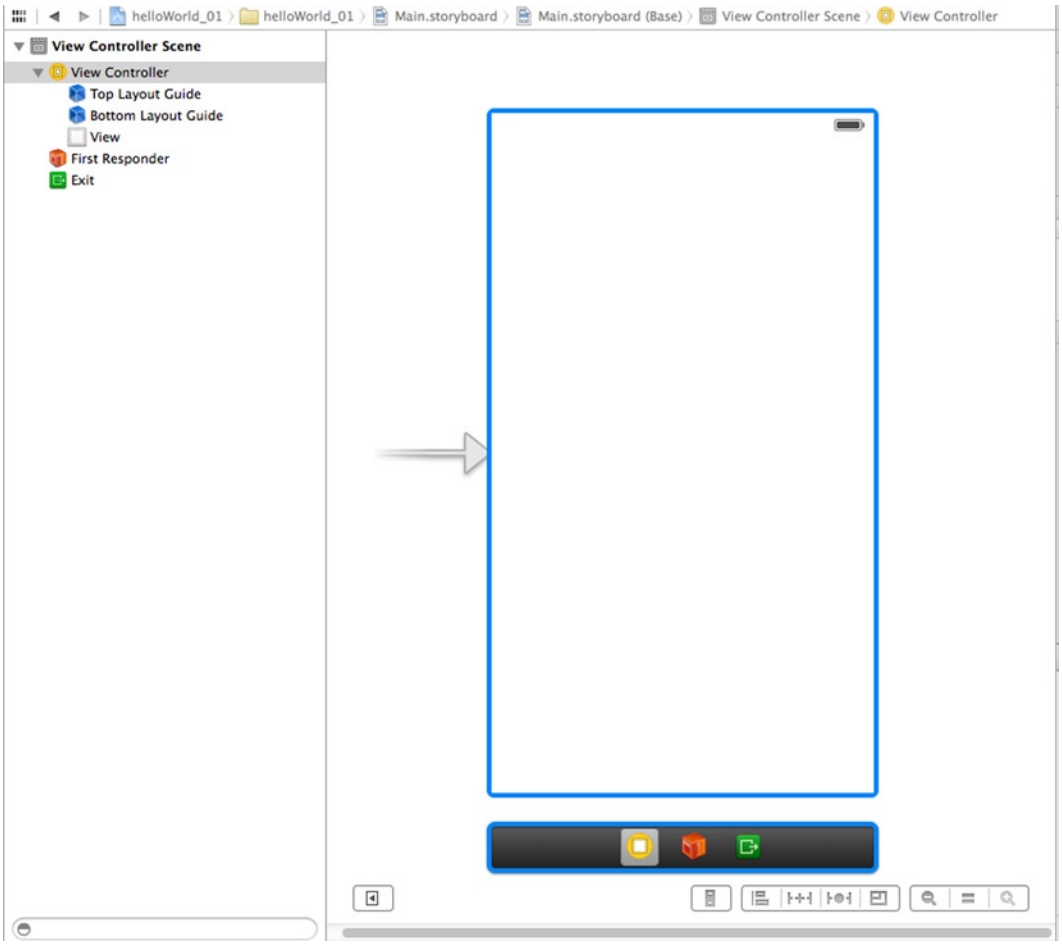

Figure 2-6. *Exploring the scene in interface builder*

To the right of this, you *should* see the utility area, where all the settings for your user interface are kept. If you don't see it, make sure that you click the "View Utility Area" button on the very top-right-hand side. It looks like a square with a shaded area to the right.

You can see the utility area in Figure 2-7.

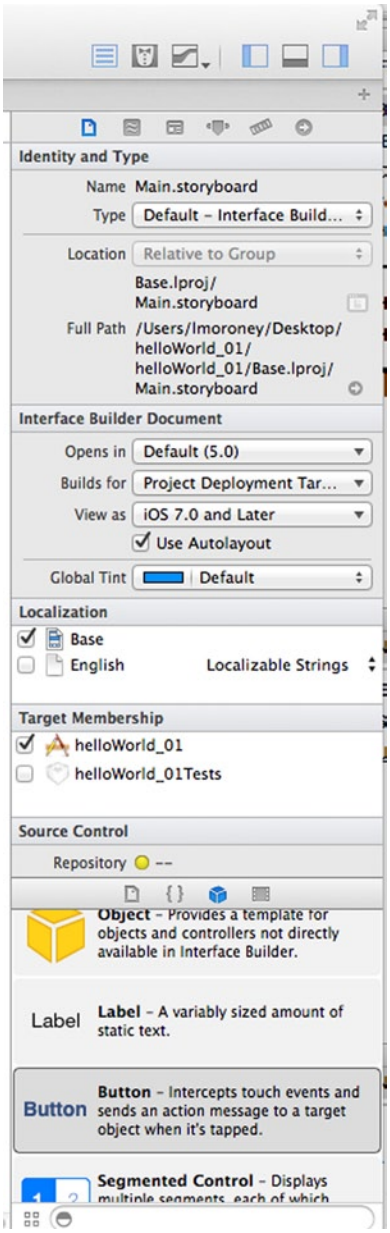

Figure 2-7. *The utility area*

Across the top of the utility area, you'll see six buttons. These are shown in Figure 2-8.

Figure 2-8. Utility area buttons

From left to right, these buttons are:

File Inspector: Used to set the details including the name, document, language, and localization for the interface

Quick Help Inspector: Gives context-sensitive help on the currently selected object

Identity Inspector: Provides details on the class underlying the view

Attributes Inspector: Displays metadata about the currently selected view, such as background color

Size Inspector: Shows definitions about how controls should stretch to fit the view

Connections Inspector: Shows definitions about how the view connects to code

Again, don't worry if you don't understand most of this yet. As you work through this book you'll be using each of these tabs, so it's good to take a quick tour to get a feel for what the controls do.

The important setting that we want you to see here is that which connects the interface that you are designing to the underlying code that you'll write. With "View Controller" selected (see Figure 2-6), make sure the "Custom Class" button is clicked, and you'll see that this user interface will map to a specific class—in this case ViewController. That class is implemented using the `ViewController.m` and `ViewController.h` files.

At the bottom of the utility area, no matter which button is selected, you'll see a list of controls. Scroll this list until you see the "Label" control, as shown in Figure 2-9.

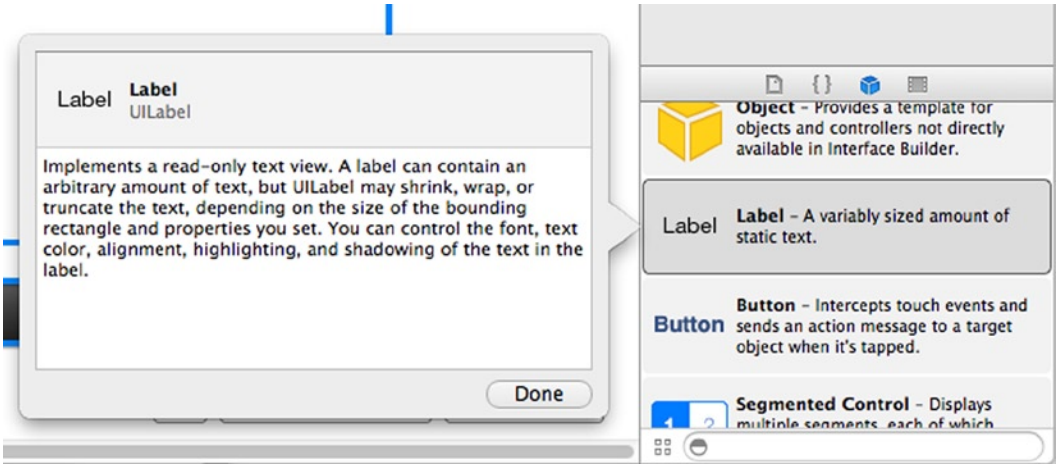

Figure 2-9. Selecting the "Label" control

Using your mouse, drag the “Label” control onto the view. As you move it around, you should see blue dotted guide lines that will help you place it. Vertical ones show you the horizontal center of the view and vice versa. Others show you borders around the edge of the screen that you might want to avoid. You can drop the label wherever you like on the view. When you’re done, it will look something like Figure 2-10.

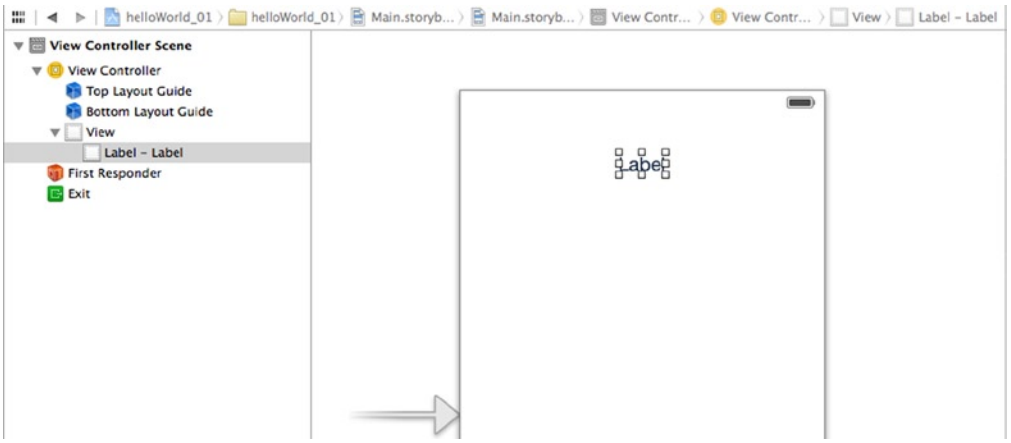

Figure 2-10. *Placing your label on the view*

You should then drag the little white dots around the label to resize it. Xcode will crop any text that goes into a label to the size of the label by default, so your “Hello, World!” text will look something like “He . . .” unless you make the label larger.

Experiment until you get the width that you like; you’ll notice that the word “Label” appears on the left of the screen. To re-center it, don’t move the label—find the “Alignment” setting in the attributes inspector and set it to “Center.”

Go back to the controls list and scroll until you see the “Button” control. Select it and drag and drop it onto the view, just like you did with the label. When you’re done, your view should look something like Figure 2-11.

Figure 2-11. *Placing the “Button” control on the view*

With the button selected (it will be highlighted on the list on the left and will have dots around it on the designer to the right), select the "Attributes Inspector" button. You'll see the "Title" property that says "Button." You can use this to change the title to "Press Me!" See Figure 2-12. (Alternatively, you could double click on the button in the designer and just type the new caption.)

***Figure 2-12.** Updating the button title*

You've now finished with the designer, so save your work before you go to the next step. You can do this by pressing ⌘s or selecting "Save" on the "File" menu.

At the top of the screen, where you opened the utility area, you should now close it and then open the "Assistant" editor. (This is the second icon from the left looking like a tuxedo.) See Figure 2-13.

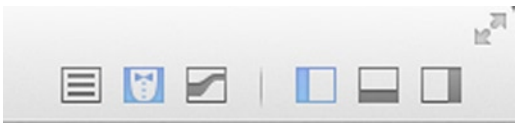

***Figure 2-13.** Opening the assistant*

Note that when an icon is selected it's shaded in light blue. So, in Figure 2-13 you can see that the utility area at the right has been closed, and that the assistant has been opened.

You should see that a code window will open beside the designer.

> **Note** If the code in the assistant window on the right is for `ViewController.m`, as shown in Figure 2-14, you'll need to make some changes. This is actually the *wrong* file, but it is often the default file that you get. If this happens, find where it says `Automatic > ViewController.m` at the top of the assistant window. You can see it in Figure 2-15. Once you've done this, click on `ViewController.m`, and a drop-down menu will appear that you can use to pick `ViewController.h`.

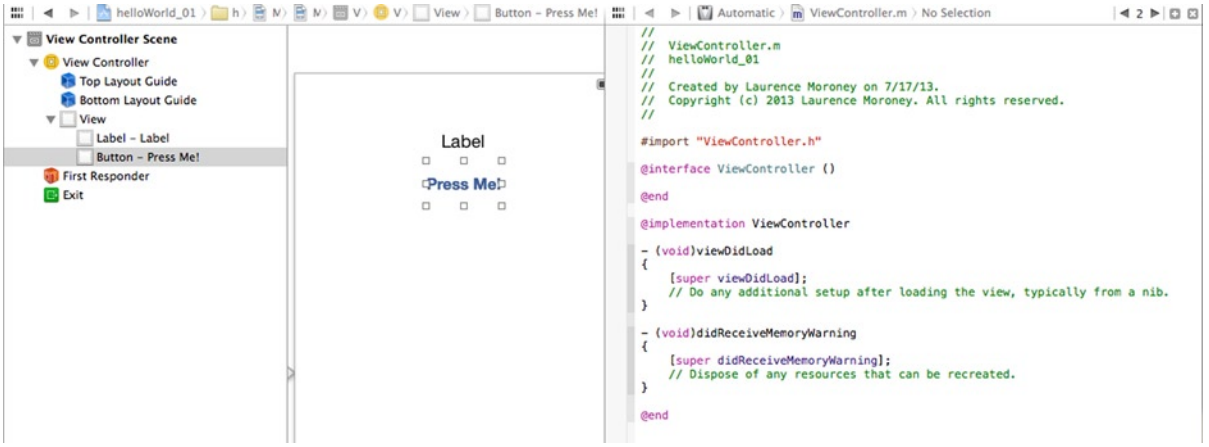

Figure 2-14. Using the assistant window

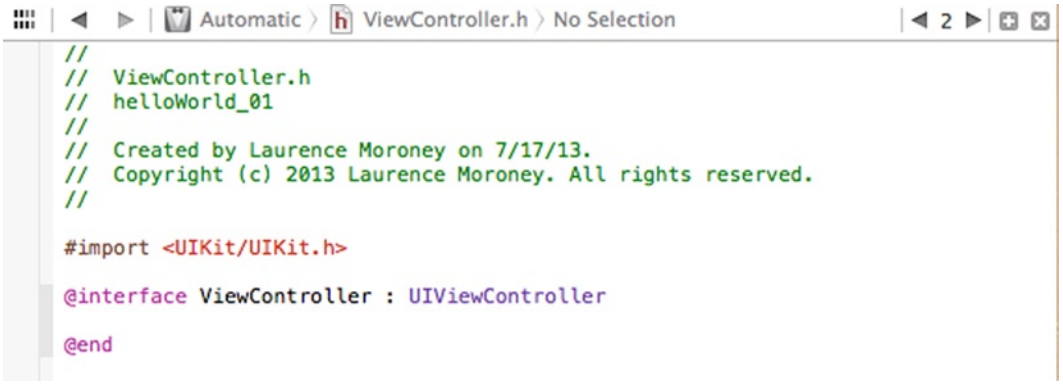

Figure 2-15. Assistant with ViewController.h selected

As mentioned earlier, classes are defined using two files. The .h (or header) file contains all the information about the class, including the names of the functions and variables used within it. The .m (or implementation) file contains all the logic. To map the controls to functions in the class, you'll need to edit the .h file, so it should be the one selected in the assistant view before you go any further.

When the user presses the button, you want something to happen. You'll write the code for this; Objective-C and iOS call this an *Action*. So, to create an action that happens in response to the user pressing your button, you should hold the CONTROL key and drag the button onto the code window, just below the word `@interface`. This creates what is called a *connection*.

You'll see a little helper window pop up. It will look like Figure 2-16.

Figure 2-16. *Connection helper window*

The default type of connection is called an *Outlet*. You'll see what that is for a little later in this chapter. What you want to create right now is an action, so change the "Connection" setting to this. You'll see that the settings change. See Figure 2-17 for how they will look now.

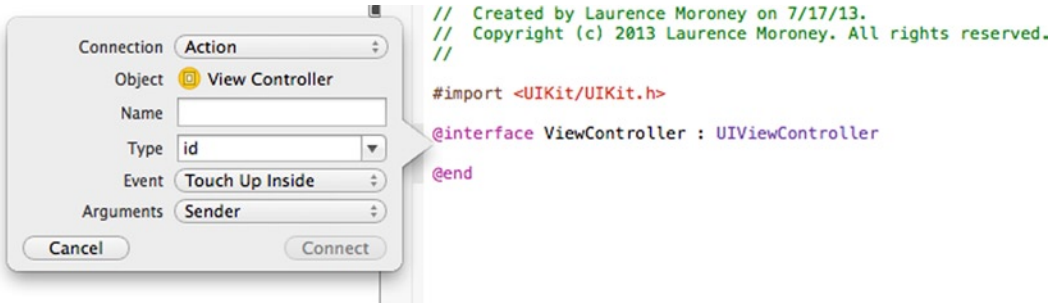

Figure 2-17. *Connection helper window for actions*

The important parts you want to set up here are the "Name" and "Event" types.

The Name setting is the name of the function that will run when the action happens. Type "btnPressed" into the Name field.

The Event setting is the type of interaction from the user that will trigger this action. Every control has a different set of events that it supports. For a button, "Touch Up Inside" is the default event. It fires, as the name suggests, when the user releases their touch inside the button.

Enter these values ("btnPressed" for Name and "Touch Up Inside" for Event), and click the "Connect" button. You'll see that the code in the assistant window has changed. See Figure 2-18. (If you inspect the .m file, you'll see that code has been added there too. You'll be editing that code in the next section.)