Bear Cahill

Covers iOS 6

iOS IN PRACTICE

Includes 98 Techniques

SAMPLE CHAPTER

MANNING

iOS in Practice

by Bear Cahill

Chapter 1

brief contents

1

Getting started with iOS development

This chapter covers

- Xcode and Objective-C
- Getting to know Xcode
- A Hello World example

I've been developing professionally for over 20 years in about every language and platform, but I believe iOS development is some of the most exciting, fun, gratifying, and challenging work I've ever done. I love iOS development.

Not only is it appealing from a developer's standpoint, it's also the leading mobile platform. This means that there's lots to do, with lots of growth and changes, and plenty of support out there from Apple, forums, other developers, books, conferences, and so on.

With the growth of iOS and other mobile platforms, tablets, which nicely bridge traditional computers and smart phones, are now a huge market. These mobile devices allow for more opportunities for development, and iOS lets you develop for both platforms simultaneously.

In this chapter, we develop an iOS application (or *app*). We need to go over a few topics, including setting up the development environment, but by the end of the chapter, you'll have your first app. Let's go!

1.1 The iOS development environment

Xcode is the primary tool for developing iOS (and OS X) applications. It's free from Apple and helps with a variety of development-related tasks including user interface (UI) design/development, revision control, and more.

The primary language that iOS is developed in is called *Objective-C*. Objective-C is a descendent of C, which means that all C code will compile and run in Objective-C. But, unlike C, Objective-C is object-oriented. If you know C++, Java, or other object-oriented languages, you'll have no problem understanding this language. Keep in mind that the purpose of this book isn't to teach you Objective-C, so if you find you're having a hard time with the language, you may want to take some time and use other resources to research Objective-C.

Apple also provides a rich set of frameworks. Some are required for any app and are automatically included. The rest are optional, depending on your preferences, and can greatly add to your project. When iOS first came out, displaying a map of a location was difficult and labor intensive. Adding pinned locations to the map was even more complex. When MapKit was introduced, adding a map and displaying the user's location became practically effortless.

WebKit, StoreKit, MediaPlayer, Social, and CoreData are a few more frameworks that bring ease of functionality when added to your projects. Many open source and/or third-party frameworks are available to keep you from having to reinvent the wheel for common—but complicated—functionality.

iOS development also relies heavily on the *Model-View-Controller (MVC)* architecture pattern. MVC is the separation of your development into three aspects: model, view, and controller. The *model* is the data layer (for example, the database in a project). The *view* is the UI that the user interacts with. The *controller* is between the view and the model, and it translates the user interaction to logic and accesses data as necessary.

As you can see, Xcode does a lot to facilitate what you need to do as a developer as well as enables you to do it in a fashion best suited for iOS projects. Let's look into the details of getting, installing, and becoming familiar with Xcode, and then you'll develop your first app.

1.2 Using Xcode

As stated in the previous section, Xcode is the primary development tool for iOS projects. In this section, we look at how to get Xcode from Apple and tour the various parts of Xcode to simplify iOS development.

1.2.1 Getting Xcode

Using the App Store from Apple, you can search for Xcode and quickly find it. It's free, so just click on the FREE button to begin the install (see figure 1.1). It's a large download and might take a while, but the download process is pretty easy. Xcode and related apps will then be installed in /Developer/Applications and key apps will be added to the Developer folder in Launchpad.

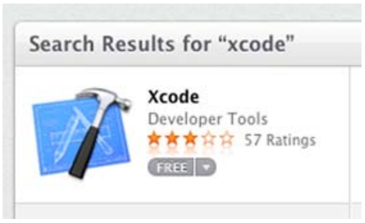


Figure 1.1 Xcode in the App Store

You may also go to http://developer.apple.com and download Xcode if you'd like to go the more manual route. There you'll also see information about joining the various developer programs such as Safari, iOS, and Mac.

In most cases, the developer programs cost money to join, but they also allow access to advance/beta releases, developer forums, and other resources. If you intend to do much iOS development, I highly recommend joining. If you intend to release any apps, you have to join.

Now that you have Xcode installed, let's look at the various aspects of it.

1.2.2 Tour of Xcode

Xcode can handle all of the major aspects of project development for iOS projects. It can manage the organization of code, linking frameworks, UI design, editing, projects (such as regular and pro versions of the same code base for separate apps), building, testing, and submission to Apple for review. In this chapter, we look at some of the basics to get started with Xcode. In later chapters, we explore more details and areas of Xcode.

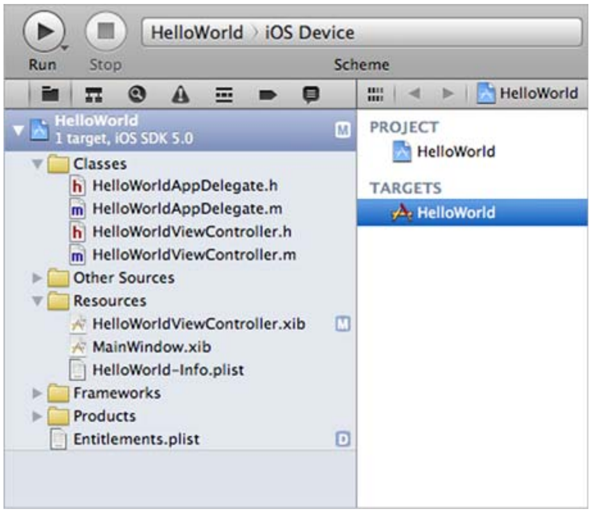
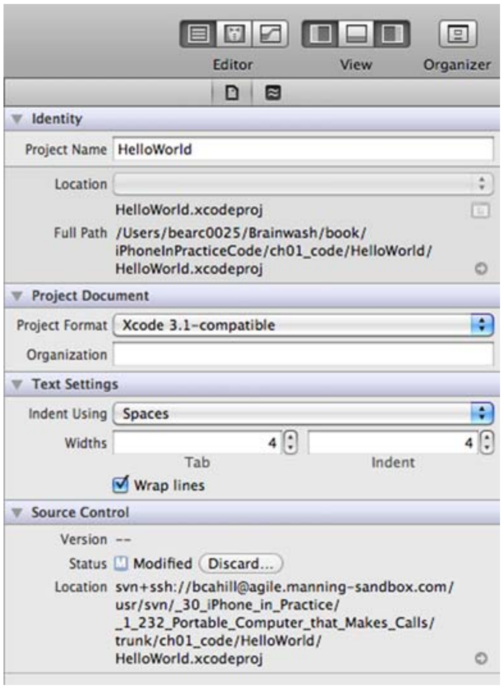


Figure 1.2 Navigator in Xcode

Given that Xcode helps in so many ways, it makes sense that there are a lot of areas, panes, views, and such included in it. The Navigator on the left displays the various files, frameworks, projects, and related items included in your project (see figure 1.2). This allows you to select files to edit or control in various ways.

The Utilities area, displayed by the right button above View on the top right, shows various aspects and settings of a selected item such as a file (see figure 1.3). Here you can see how a given item relates to other items, set various attributes, and more. This is particularly helpful when using the Interface Builder (IB) UI editor to set attributes on visual items.

The Editor is probably the most familiar-looking item in Xcode because all development needs a way to edit code (see figure 1.4). But the Editor serves as the editor not only for code, but also the UI and data (such as database design for CoreData), which you'll see throughout the projects in this book.



Identity

Project Name HelloWorld

Location

HelloWorld.xcodeproj

Full Path /Users/bearc0025/Brainwash/book/iPhoneInPracticeCode/ch01_code/HelloWorld/HelloWorld.xcodeproj

Project Document

Project Format Xcode 3.1-compatible

Organization

Text Settings

Indent Using Spaces

Widths 4 4

Tab Indent

✓ Wrap lines

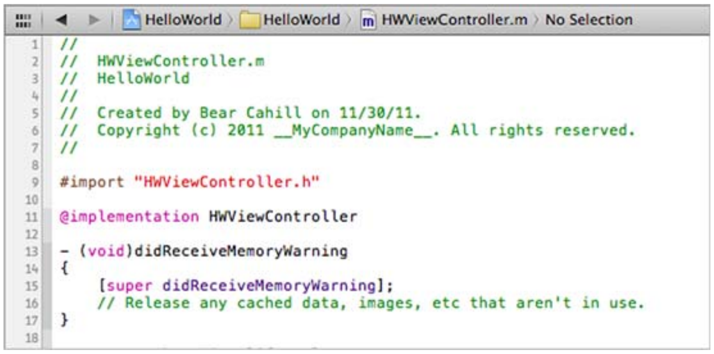
Source Control

Version --

Status Modified Discard...

Location svn+ssh://bcahill@agile.manning-sandbox.com/usr/svn/_30_iPhone_in_Practice/_1_232_Portable_Computer_that_Makes_Calls/trunk/ch01_code/HelloWorld/HelloWorld.xcodeproj

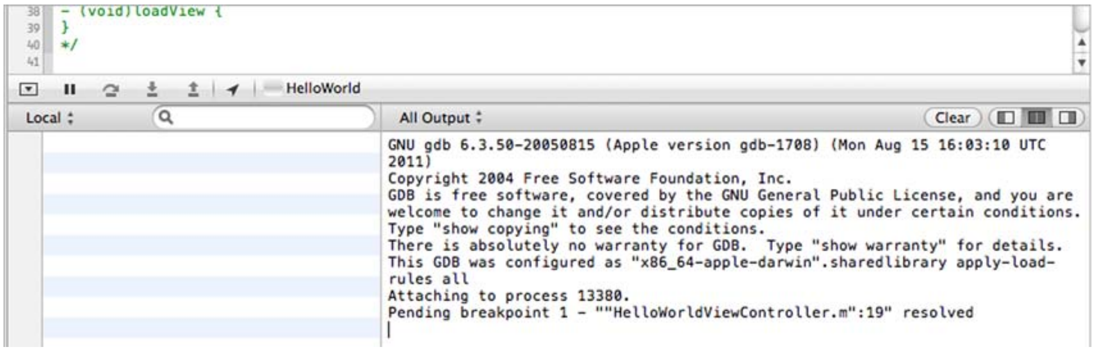
Figure 1.3 Utilities in Xcode for HWViewController.m



```
//
//  HWViewController.m
//  HelloWorld
//
//  Created by Bear Cahill on 11/30/11.
//  Copyright (c) 2011 __MyCompanyName__. All rights reserved.
//

#import "HWViewController.h"

@implementation HWViewController

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Release any cached data, images, etc that aren't in use.
}
```

Figure 1.4 Xcode Editor with HWViewController.m selected

The Debug area displays at the bottom and can be split to display the Console on the right for viewing standard output (see figure 1.5). Both of these can be helpful for displaying variable values and output during testing.



```
- (void)loadView {
}
*/
```

```
GNU gdb 6.3.50-20050815 (Apple version gdb-1708) (Mon Aug 15 16:03:10 UTC
2011)
Copyright 2004 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "x86_64-apple-darwin".sharedlibrary apply-load-
rules all
Attaching to process 13380.
Pending breakpoint 1 - ""HelloWorldViewController.m":19" resolved
```

Figure 1.5 Debug view and console during Hello World execution

The Toolbar is located at the top of the window and is helpful for displaying various areas, starting/stopping test runs, and selecting what scheme to use for building (see figure 1.6).

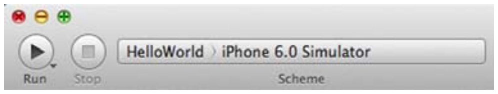
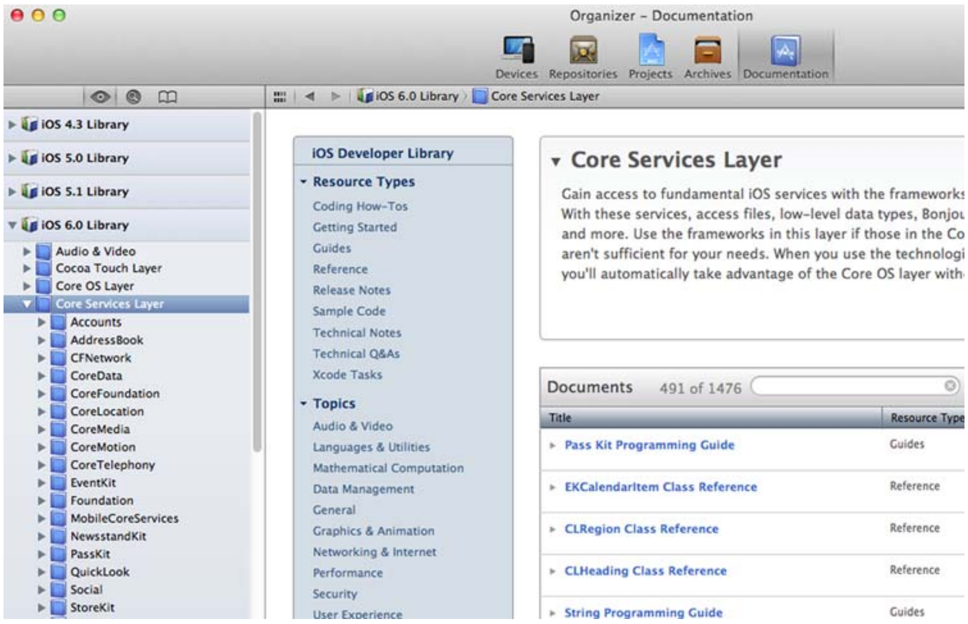


Figure 1.6 Xcode toolbar at the top of the window

The Organizer, which is displayed in the Window menu, is used for a variety of aspects of development. It displays framework and other help documentation, facilitates submitting your binary to the AppStore for review, organizes your various devices, and more (see figure 1.7). It can help you keep track of your provisioning profiles as well as give you access to your crash reports on your devices (not that your apps will crash—other people need this).



Core Services Layer

Gain access to fundamental iOS services with the frameworks
With these services, access files, low-level data types, Bonjou
and more. Use the frameworks in this layer if those in the Co
aren't sufficient for your needs. When you use the technologi
you'll automatically take advantage of the Core OS layer with

Documents 491 of 1476

Title	Resource Type
Pass Kit Programming Guide	Guides
EKCalendarItem Class Reference	Reference
CLRegion Class Reference	Reference
CLHeading Class Reference	Reference
String Programming Guide	Guides

Figure 1.7 Xcode Organizer displaying framework documentation

The Organizer can be particularly helpful in bringing up context-related documentation by using command-click on text in your code. Also, it gives you access to helpful documents like the "Apple Human Interface Guidelines" and "Learning Objective-C: A Primer." Both are recommended reading.

Now that you have your bearings with Xcode and its environment, let's build that app!

1.3 A quick Hello World app

As a way to explore Xcode more and get your feet wet in iOS development, you'll create a basic app. It won't do much, but it will be a quick pass through the basics of creating an app.

First, you'll create a new project that includes several steps to specify necessary aspects of your project. Then you'll create the UI for your app and run it.

1.3.1 Creating a New Project

Start Xcode and, when prompted, select Create a New Project (see figure 1.8).

You'll be presented with various options for a template for your project. Be sure that Application is selected under iOS on the top left. The appropriate options will be displayed on the right. Select Single View Application (see figure 1.9) and click Next on the bottom right.

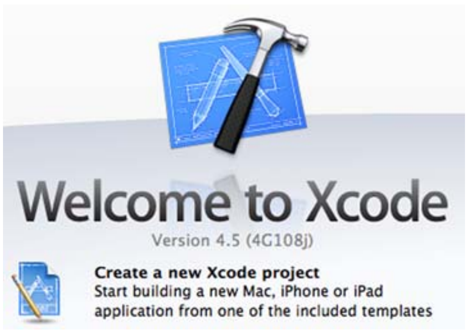
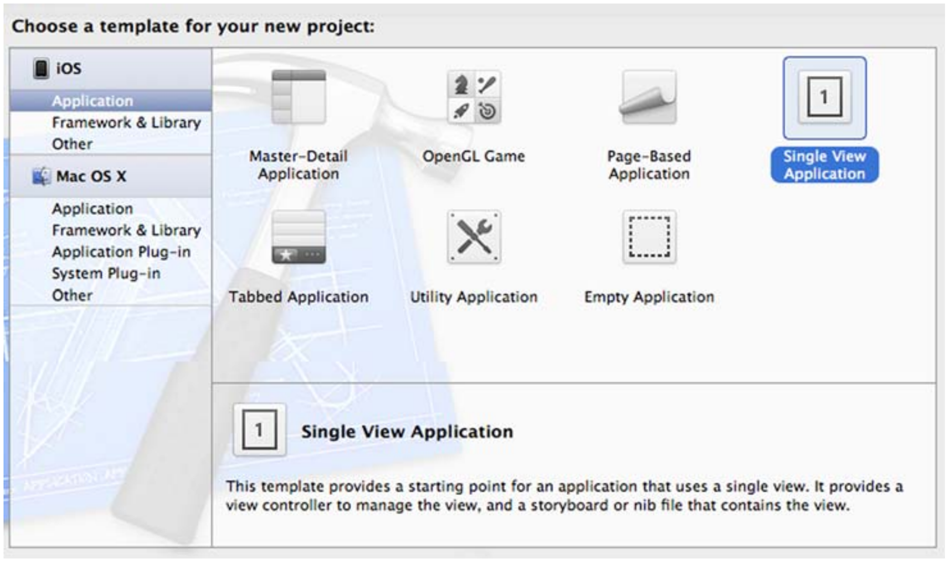

Welcome to Xcode
Version 4.5 (4G108j)
Create a new Xcode project
Start building a new Mac, iPhone or iPad application from one of the included templates

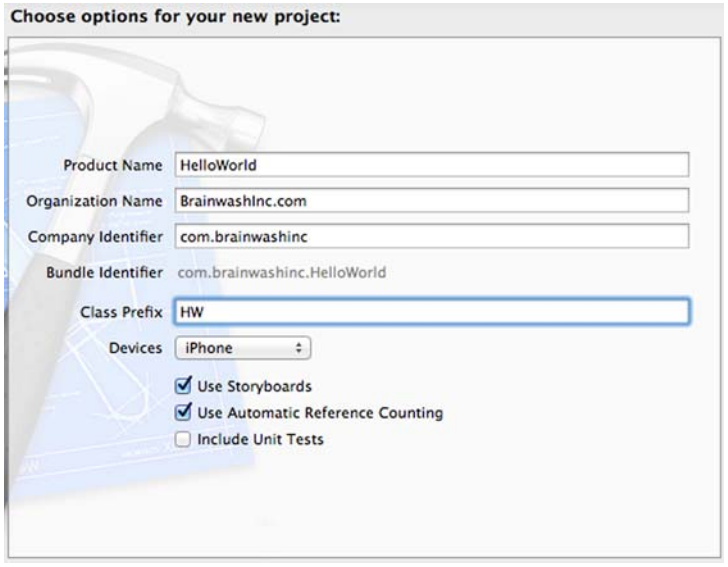
Figure 1.8 Create a new project with Xcode.



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 1.9 Single View Application template for new project

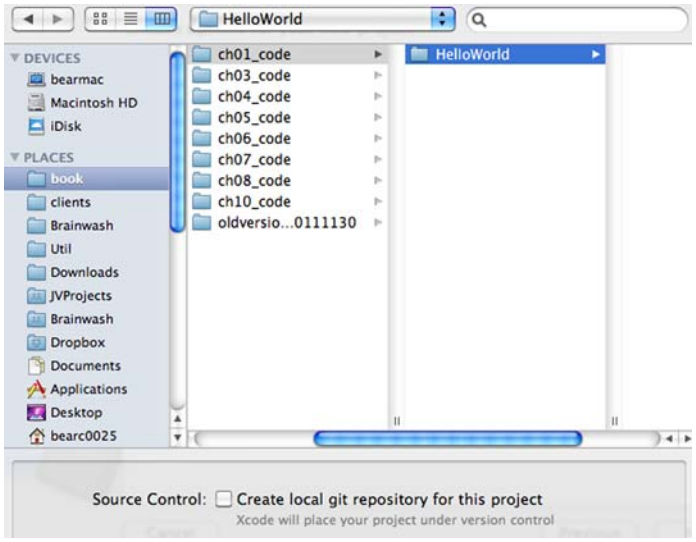
You'll then be prompted to name your product as well as set the company identifier, which is typically a reverse DNS value. You'll also specify a class prefix (for the naming convention) and specify the device family (such as iPhone). Finally, select Storyboard for UI design, reference counting for memory management, and unit tests (see figure 1.10).


Choose options for your new project:

Product Name: HelloWorld
Organization Name: BrainwashInc.com
Company Identifier: com.brainwashinc
Bundle Identifier: com.brainwashinc.HelloWorld
Class Prefix: HW
Devices: iPhone
✓ Use Storyboards
✓ Use Automatic Reference Counting
☐ Include Unit Tests

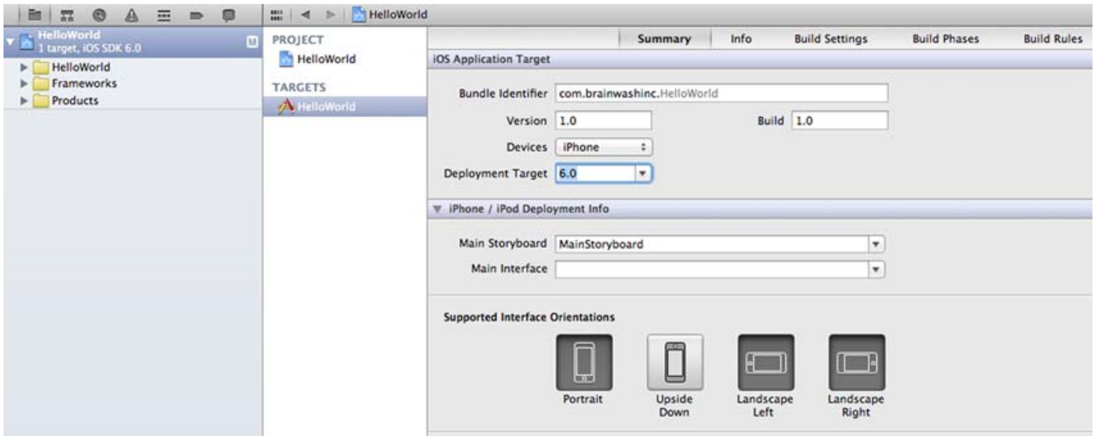
Figure 1.10 Xcode options for a new project

Click Next and you'll be presented with a Finder window to specify the location of the project (see figure 1.11). You can also create a local git repository for your project during this step (see the bottom of figure 1.11).



Source Control: ☐ Create local git repository for this project
Xcode will place your project under version control

Figure 1.11 Specify the location of a new project.



iOS Application Target

Bundle Identifier: com.brainwashinc.HelloWorld
Version: 1.0
Build: 1.0
Devices: iPhone
Deployment Target: 6.0

iPhone / iPod Deployment Info

Main Storyboard: MainStoryboard
Main Interface:

Supported Interface Orientations

Figure 1.12 Xcode Target summary for new project

Click Create and your project is now ready to develop! Xcode will display your default Target's summary (see figure 1.12). You'll be able to see the selections you made. Pay particular attention to the naming convention using the prefix in the Navigator.

Note also the Main Storyboard setting of MainStoryboard because you chose to use the Storyboard setting during your app creation. If you hadn't checked that box, this setting would be empty and you'd have a Main Interface setting instead.

The default Main Interface settings would relate to a UI design file with the file extension of xib. Instead, you have a .storyboard file in your project. Most of the projects in this book use XIB files for UI design, but you'll use Storyboard here. Now let's look at the file and the UI of your first app.

1.3.2 *Editing the user interface*

Before you change your project based on the template, let's run it. Yep, it's already in a state where you can compile and run it. Make sure the iPhone Simulator is selected in the scheme pull-down menu on the top left (see figure 1.13) and click Run.

Xcode will compile, link, and execute the code using the iOS Simulator. It will only display a blank white screen because your app doesn't do anything yet. Let's change that!

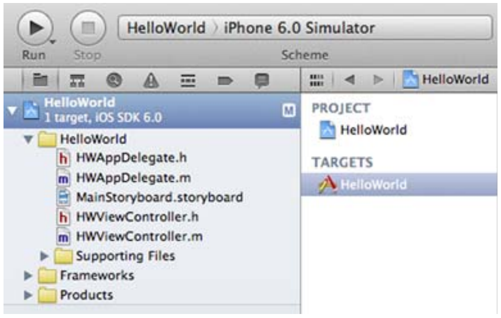


Figure 1.13 Xcode scheme selected as an iPhone Simulator

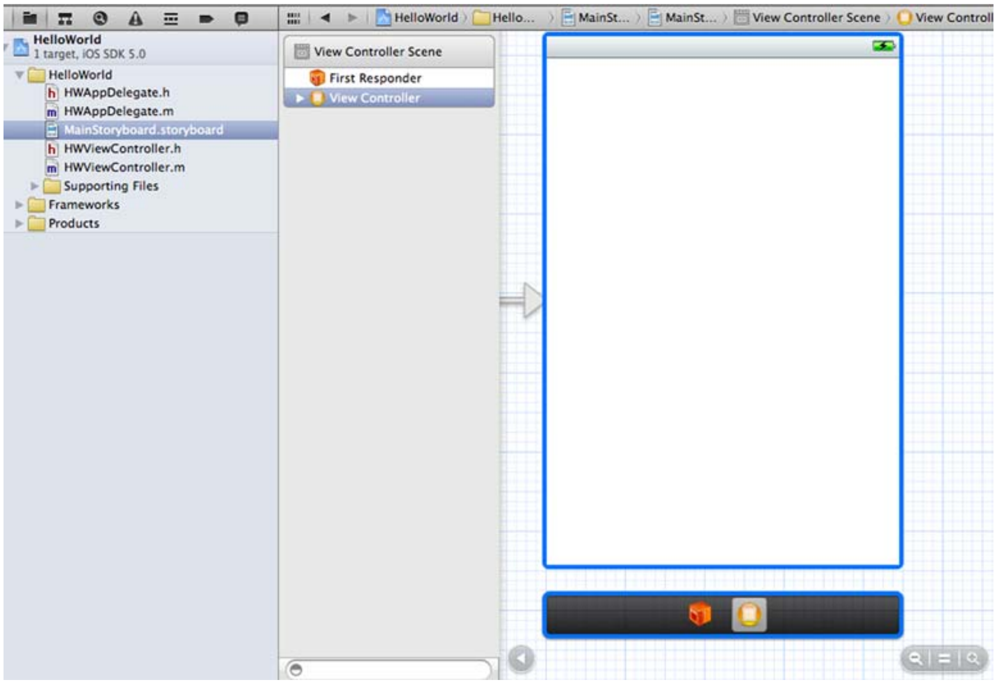


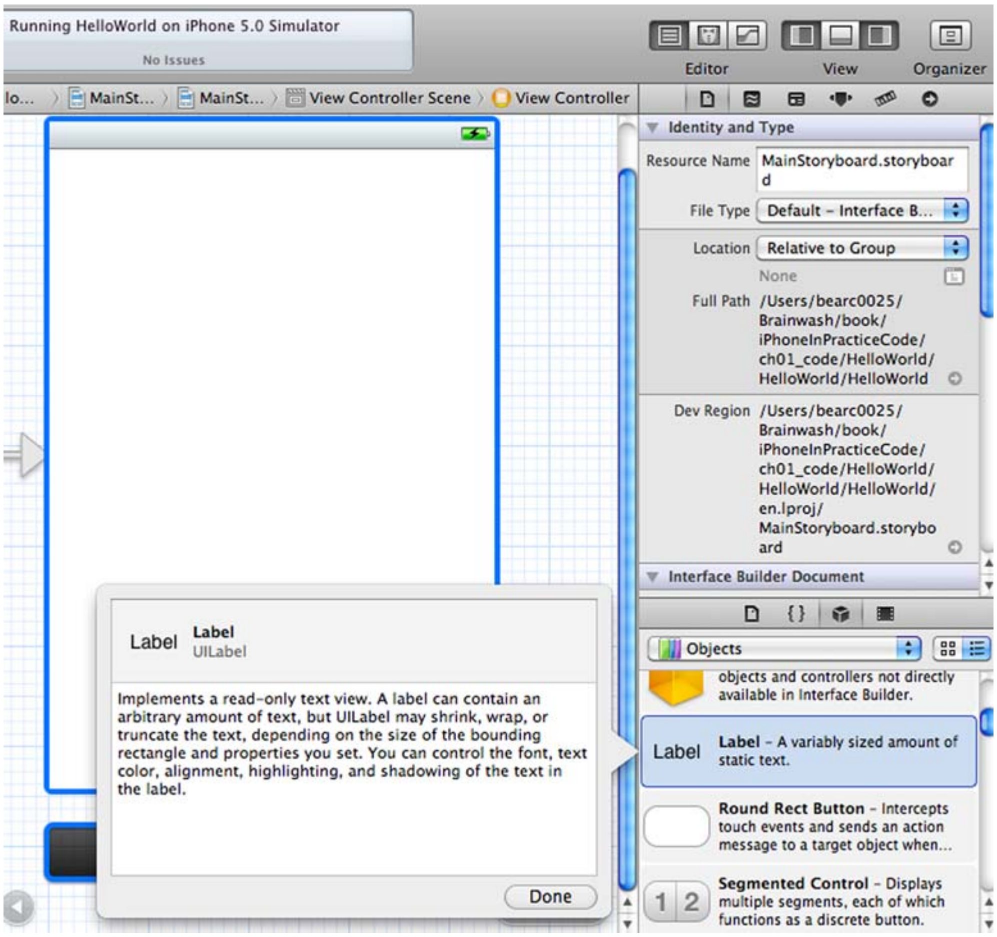
Figure 1.14 Storyboard file displayed in the Editor

Click on the Storyboard file in your project (for example, MainStoryboard.storyboard); the UI contents will be displayed in the Editor, which, in this case, is Interface Builder for the user interface (see figure 1.14). That white rectangle is the view controller you saw displayed when you ran the project in the simulator.

Note the HWViewController.h/.m files in the Navigator. Note the items in the list on the left in the Editor (see figure 1.14). The one listed as View Controller is an instance of your `HWViewController` class. Therefore, changes to it in the Editor will affect how your app runs.

Be sure that Utilities is visible (right button above View on the top right of Xcode) and note that there's a list of items at the bottom (see figure 1.15).

That list on the bottom right contains other UI items that can be dropped onto your UI. Scroll down the list until you see Label. Drag it into the Editor and drop it in the big white area of your view controller.


Running HelloWorld on iPhone 5.0 Simulator
No Issues
Editor
View
Organizer
lo… MainSt… MainSt… View Controller Scene View Controller
Identity and Type
Interface Builder Document
Objects
Done

Resource Name MainStoryboard.storyboard

File Type Default - Interface B…

Location Relative to Group

None

Full Path /Users/bearc0025/Brainwash/book/iPhoneInPracticeCode/ch01_code/HelloWorld/HelloWorld/HelloWorld

Dev Region /Users/bearc0025/Brainwash/book/iPhoneInPracticeCode/ch01_code/HelloWorld/HelloWorld/en.lproj/MainStoryboard.storyboard

Label **Label** UILabel

Implements a read-only text view. A label can contain an arbitrary amount of text, but UILabel may shrink, wrap, or truncate the text, depending on the size of the bounding rectangle and properties you set. You can control the font, text color, alignment, highlighting, and shadowing of the text in the label.

objects and controllers not directly available in Interface Builder.

Label **Label** - A variably sized amount of static text.

Round Rect Button - Intercepts touch events and sends an action message to a target object when…

1 2 **Segmented Control** - Displays multiple segments, each of which functions as a discrete button.

Figure 1.15 Utilities displayed for Storyboard file

Double-click on the newly dropped label and type `Hello World` (see figure 1.16). Now run your app again and … congrats! You have a functioning app! Check out figure 1.17 to see what your app's label should look like.

Figure 1.16 Adding a label to the project UI

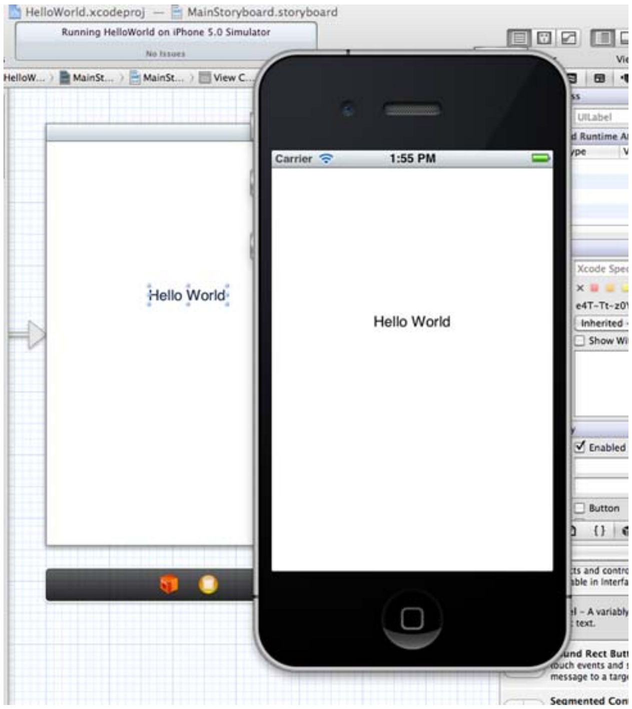

HelloWorld.xcodeproj — MainStoryboard.storyboard
Running HelloWorld on iPhone 5.0 Simulator
No Issues
Carrier
1:55 PM
Hello World
Hello World

Figure 1.17 Hello World app running in iOS Simulator

Now run your app again and … congrats (see figure 1.17)!

1.4 Summary

Great job! Your first app is done! Not so painful, right? Now you're an iOS developer (although I don't expect you to make much money from the app you created)!

In this chapter, you learned about Xcode, including how to get it and its various parts, areas, views, editors, and so on. Based on that knowledge, we quickly moved into developing a Hello World app to get a taste of iOS development.

But that's only the start. I'm sure you've seen the amazing things that iOS apps can do, so you know there's so much more possible than what you've done so far. In the next chapter, you'll take the next step of including user interaction in your Hello World app. There's no limit to what you can do, so let's dive in!

iPHONE/iPAD DEVELOPMENT

iOS IN PRACTICE

Bear Cahill

When you are building an iOS app, you want more than basic concepts—you want real answers to practical problems. You want iOS in Practice.

This book distills the hard-won experience of iOS developer Bear Cahill into 98 specific iOS techniques on key topics including managing data, using media, location awareness, and many more. And the sample apps are wonderful! As you pull them apart, you'll see two things: experienced app development and creative design savvy in action.

What's Inside

- *WhereIsMyCar* drives you through maps, CoreLocation, and camera access.
- *PlayMyLists* tunes in on settings, audio, and shake detection.
- *Rock*, *Paper*, *Scissors* explores networking, voice, in-app purchase, push notification, and invitations.
- Examples written for iOS 6 using Xcode 4.5.

Written for readers who know the basics of Objective-C and are interested in practical app development.

Bear Cahill is an independent iOS developer whose clients include both large and small companies. He has created or contributed to numerous popular apps and is a frequent speaker and presenter. He writes a blog at brainwashinc.com.

Free eBook
SEE INSERT

"The direct path to getting your app into the App Store ... hands-on and real-world."

—Jonas Bandi, TechTalk

"If you're a web developer wanting to add iOS to your repertoire, Bear is the guide for you."

—Stephen Aument
web and mobile consultant

"A quick and easy way to get into the iOS mindset."

—Christopher Haupt
Webvanta Inc.

To download their free eBook in PDF, ePub, and Kindle formats, owners of this book should visit manning.com/iOSinPractice

ISBN 13: 978-1-61729-126-5
ISBN 10: 1-617291-26-9
54499
9 781617 291265

MANNING $44.99 / Can $47.99 [INCLUDING eBOOK]