

Bear P. Cahill

ios

IN PRACTICE



- 42 TECHNIQUES
- GET IT DONE
- GET SAVVY

MEAP



MANNING



**MEAP Edition
Manning Early Access Program
iOS in Practice MEAP version 5**

Copyright 2011 Manning Publications

For more information on this and other Manning titles go to
www.manning.com

Table of Contents

1. Getting Started with iOS Development
2. Creating an iOS Application
3. Using View Controllers and Images in PicDecor
4. Accessing the Address Book/Contacts in Dial4
5. MapKit and the Camera in WhereIsMyCar
6. Settings, Audio and Shake Detection in TimeDown
7. CoreData, iPod Access And Playing Music—PlayMyLists
8. Push Notification, In-App Purchase—Rock, Paper, Scissors
9. GameCenter, Leaderboards and Achievements—Rock, Paper, Scissors
10. iTunes API, iPad and iAd—iTunes API Demo
- A. iOS Developer Programs and App Distribution

Getting Started with iOS Development



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

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

Not only that, but with the growth of iOS and other mobile platforms, tablets are now a huge market which nicely bridge between traditional computers and smartphones. These mobile devices allow for more opportunities to develop and iOS lets you develop for both platforms simultaneously.

Let's get started now and develop an iOS app in this chapter. We'll need to cover a few things including setting up the development environment, Xcode, 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 UI design/development, revision control and more.

The primary language iOS is developed in is Objective-C. Objective-C is a descendent of C which means all C code will compile and run in Objective-C. Unlike C, Objective-C is object oriented. If you know C++, Java or other object

oriented languages you'll fit right in in that respect. However, this book is not written 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 research Objective-C.

Apple also provides a rich set of frameworks. Some are required for any app and automatically included. The rest you may include depending on your project and can greatly add to your project. When iOS first came out, displaying a map of a location was quite difficult and manual. Adding pinned locations to the map was even more complex. So when MapKit was introduced, it made adding a map and displaying the user's location practically effortless.

WebKit, StoreKit, MediaPlayer, Twitter and CoreData are just a few more frameworks that bring this type of ease to a lot of functionality to add to your projects. And there are also a lot of open source and/or third party frameworks out there 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 (e.g., the database in a project). The View the the User Interface (UI) which the user interacts with. The Controller is between the View and the Model and 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 facilitating how 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 then we'll develop our 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'll look at how to get Xcode from Apple and tour the various parts of Xcode to ease our 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 so it might take a while, but it's pretty easy. Xcode and related apps will 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 developer.apple.com and download Xcode if you'd like to go the more manual route. There you will also see information about joining the various developer programs: Safari, iOS and Mac.

The developer programs, in most cases, cost money to join, but also allow you 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 we 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 (e.g., regular and 'Pro' version of the same code base for separate apps), building, testing and submission to Apple for review. In this chapter, we'll look at some of the basics to get started with Xcode. In later chapters, more details and areas of Xcode will be explored.

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.

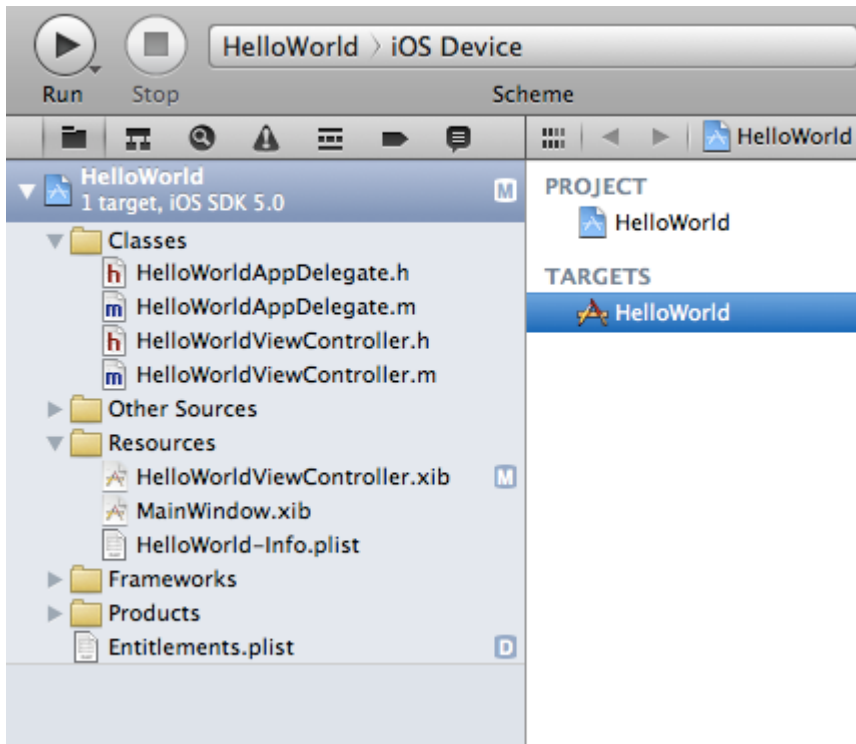


Figure 1.2 Navigator in Xcode

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

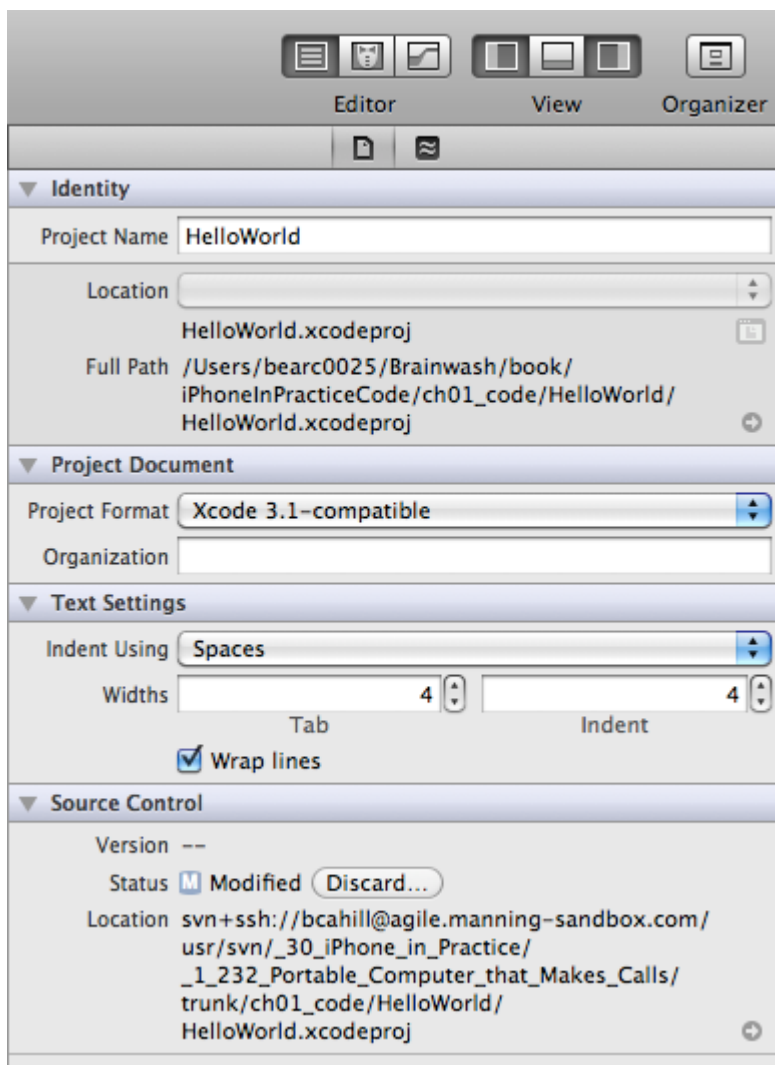


Figure 1.3 Utilities in Xcode for HWViewController.m

The Editor is probably the most familiar looking item in Xcode as just about all development needs a way to edit code (see figure 1.4). However, the Editor serves as the editor not only for code but also the UI and data (e.g., database design for CoreData) which we will see throughout the projects in this book.



```

1 //
2 // HWViewController.m
3 // HelloWorld
4 //
5 // Created by Bear Cahill on 11/30/11.
6 // Copyright (c) 2011 __MyCompanyName__. All rights reserved.
7 //
8
9 #import "HWViewController.h"
10
11 @implementation HWViewController
12
13 - (void)didReceiveMemoryWarning
14 {
15     [super didReceiveMemoryWarning];
16     // Release any cached data, images, etc that aren't in use.
17 }
18

```

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 1.4). Both of these can be very helpful for displaying variable values and output during testing.



```

38 - (void)loadView {
39 }
40 */
41

```

Local

All Output

```

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 HelloWorld 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 1.5).



Figure 1.6 Xcode Toolbar at the top of the Window

The Organizer, displayed from 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 image 1.6). 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).

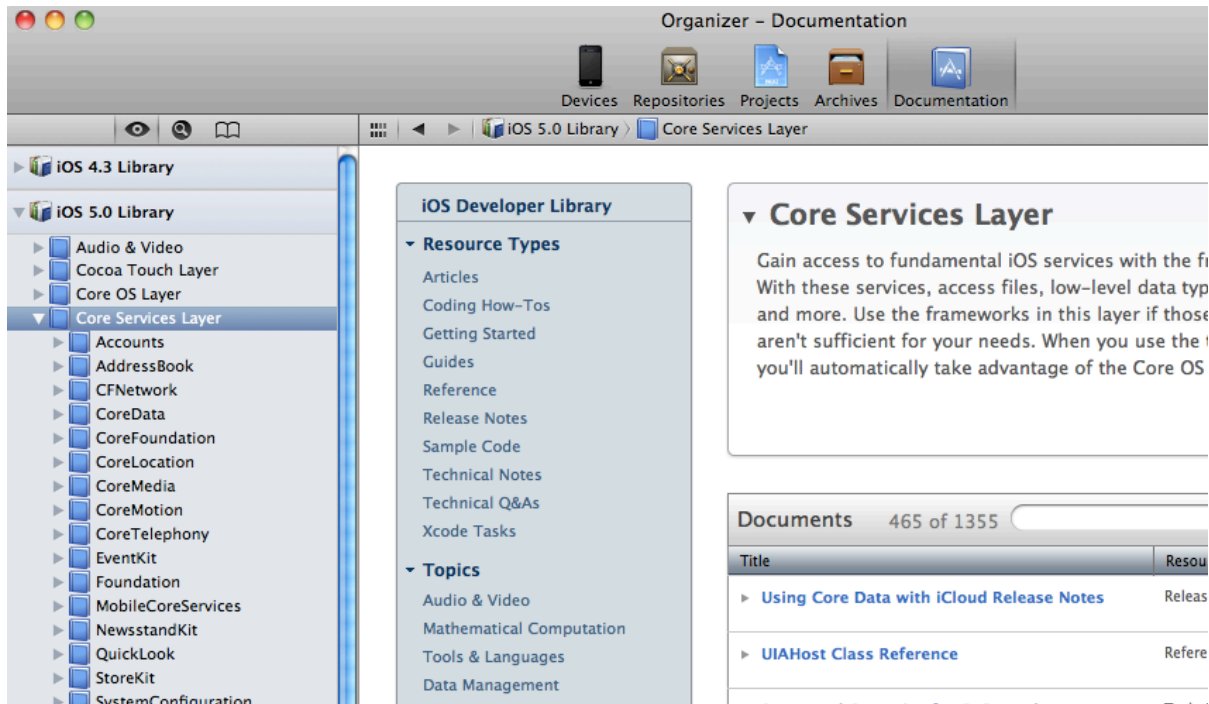


Figure 1.7 Xcode Organizer displaying Framework Documentation

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 we have our bearings with Xcode and our environment, let's build that app!

1.3 A Quick Hello World App

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

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

1.3.1

Start Xcode and, when prompted, select to create a new project (see figure 1.7).



Figure 1.8 Create a New Project with Xcode

You'll be presented with various options for a template for your project. Be sure Application is selected under iOS on the top left. The appropriate options are displayed on the right. Select Single View Application (see figure 1.8) and click Next on the bottom right.

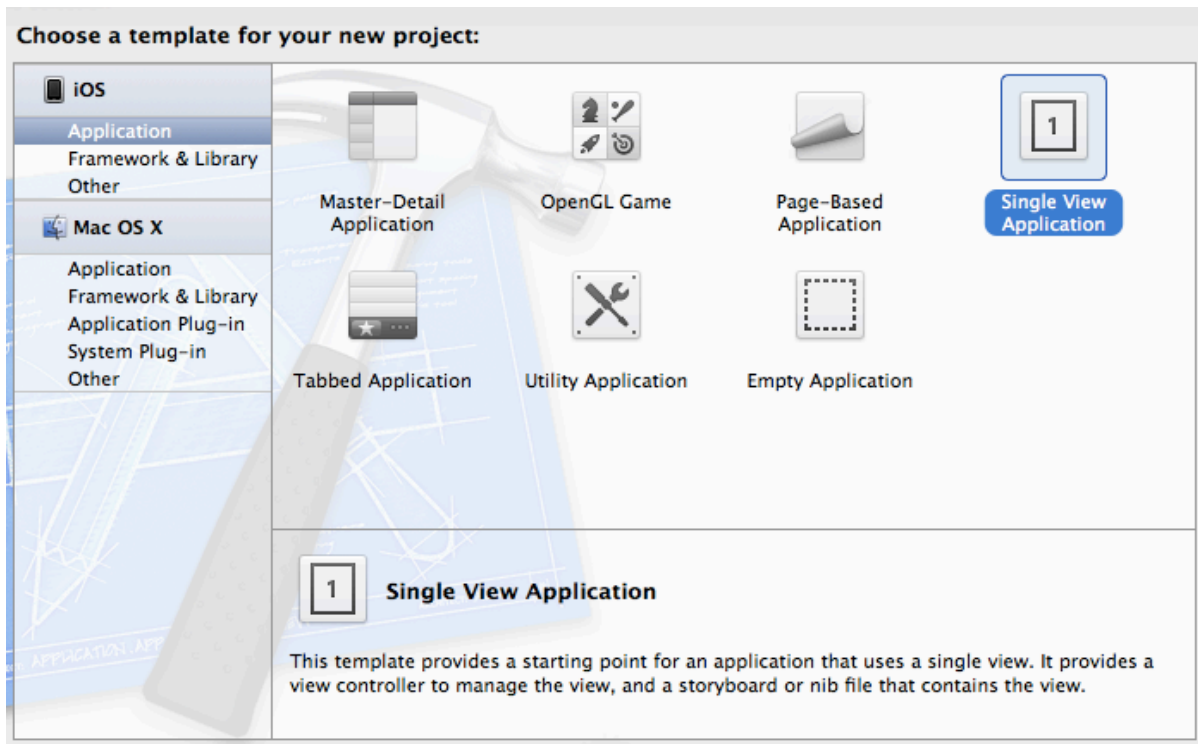
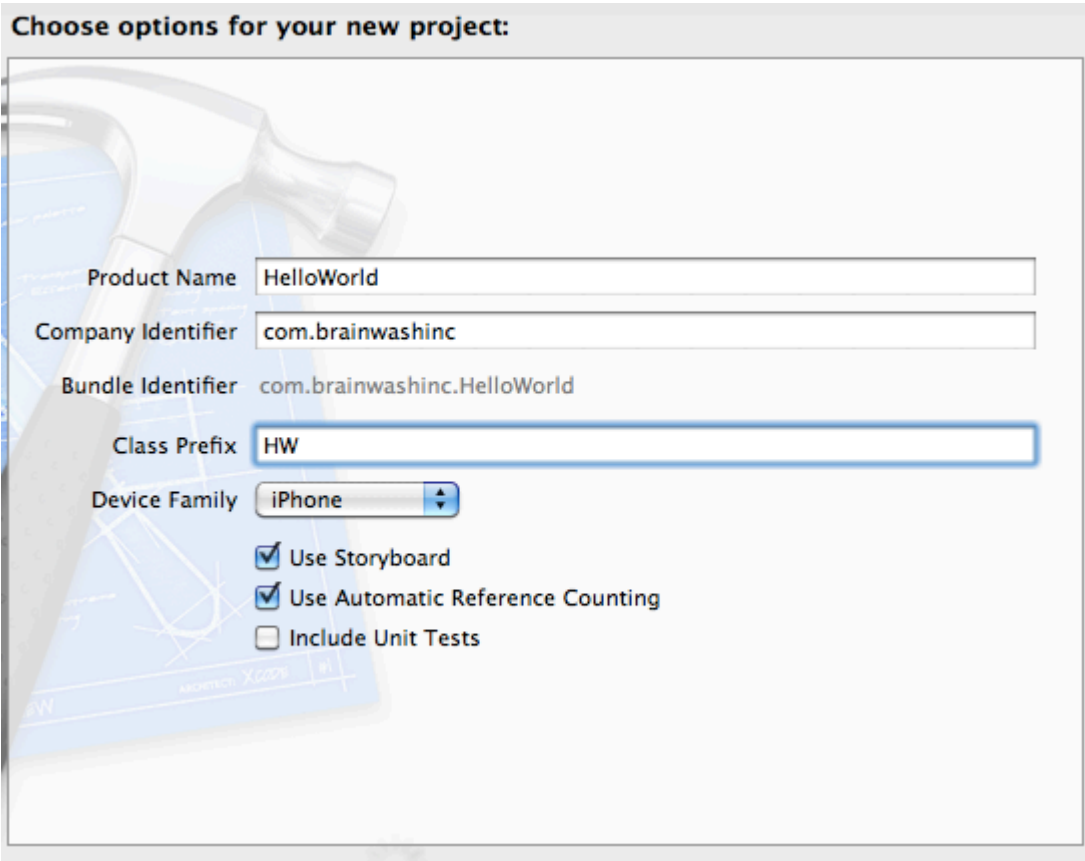


Figure 1.9 Single View Application Template for New Project

You will then be prompted to name your product as well as set the company identifier which is typically a reverse DNS value. You also specify a class prefix

(for the naming convention) and specify the device family (e.g., iPhone). Finally, you can select to use Storyboard for UI design, reference counting for memory management and unit tests (see figure 1.9).



The image shows a screenshot of the Xcode 'Choose options for your new project' dialog box. The dialog has a title bar that reads 'Choose options for your new project:'. Below the title bar is a large, faint background image of a hammer and a blueprint. The main content area contains several input fields and checkboxes:

- Product Name:** A text field containing 'HelloWorld'.
- Company Identifier:** A text field containing 'com.brainwashinc'.
- Bundle Identifier:** A text field containing 'com.brainwashinc.HelloWorld'.
- Class Prefix:** A text field containing 'HW'.
- Device Family:** A dropdown menu with 'iPhone' selected.
- Use Storyboard:** A checked checkbox.
- Use Automatic Reference Counting:** A checked checkbox.
- Include Unit Tests:** An unchecked checkbox.

Figure 1.10 Xcode Options for a New Project

Click Next and you'll be presented with a Finder window specify the location of the project (see figure 1.10). You can also create a local git repository for your project during this step (see the bottom of figure 1.10).

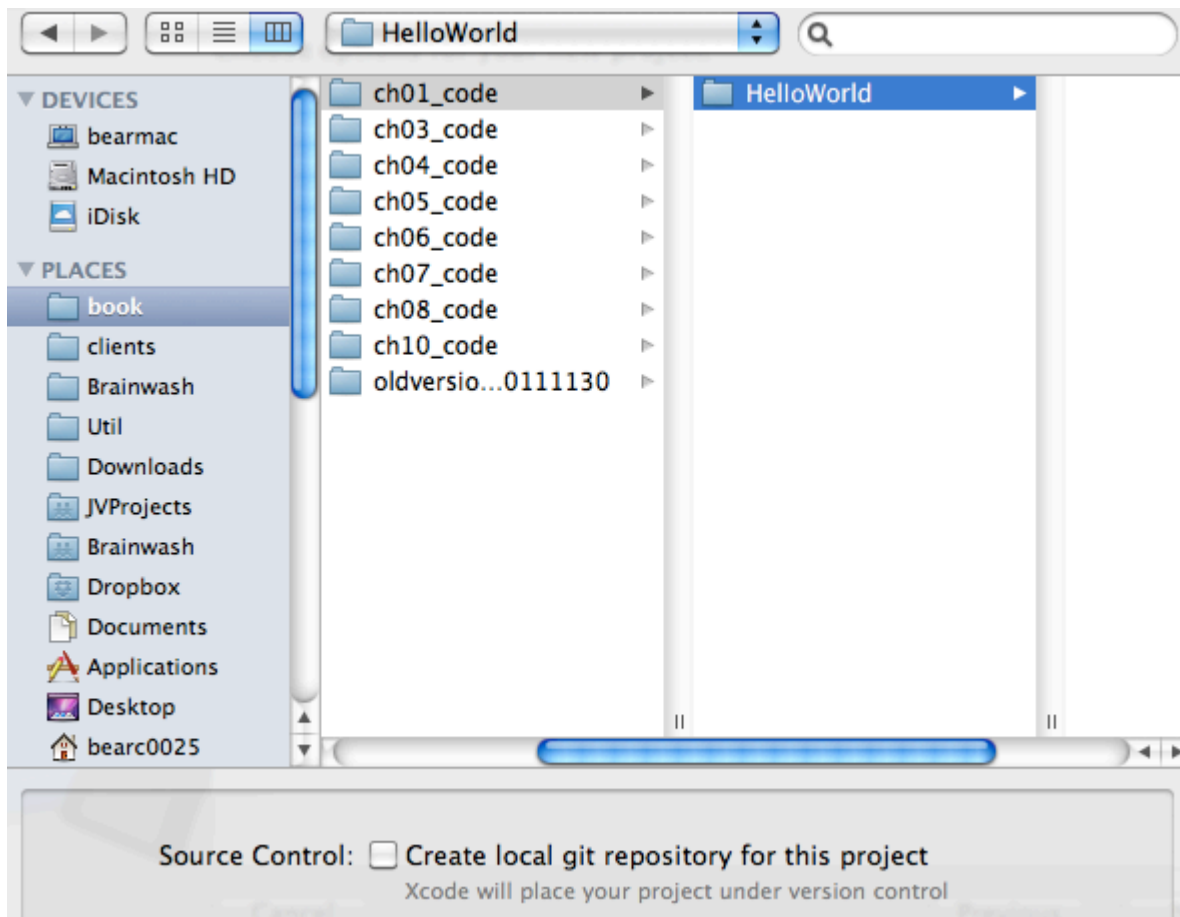


Figure 1.11 Specify the location of a New Project

Click Create and your project is created and Xcode will display your default Target's Summary (see figure 1.11). You'll notice that the selections you made. Notice in particular the naming convention using the prefix in the Navigator.

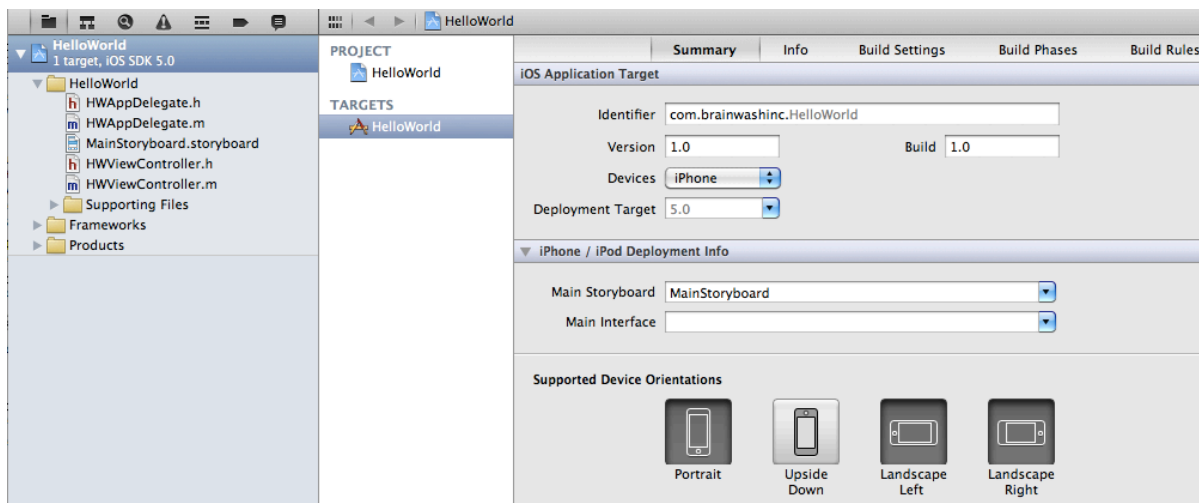


Figure 1.12 Xcode Target Summary for New Project

Notice also the "Main Storyboard" setting of MainStoryboard since we selected

to use Storyboard during our app creation. If we had not checked that box, this setting would be empty and we would 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 we have a .storyboard file in our project. Most of the projects in this book use xib files for UI design, but we'll use Storyboard here. Now let's look at that file and the UI of our first app.

1.3.2

Before we change our project based on the template, let's run it. Yep, it's already in a state we can compile and run it. Make sure the iPhone Simulator is selected in the scheme pulldown on the top left (see figure 1.12) and click the Run button.

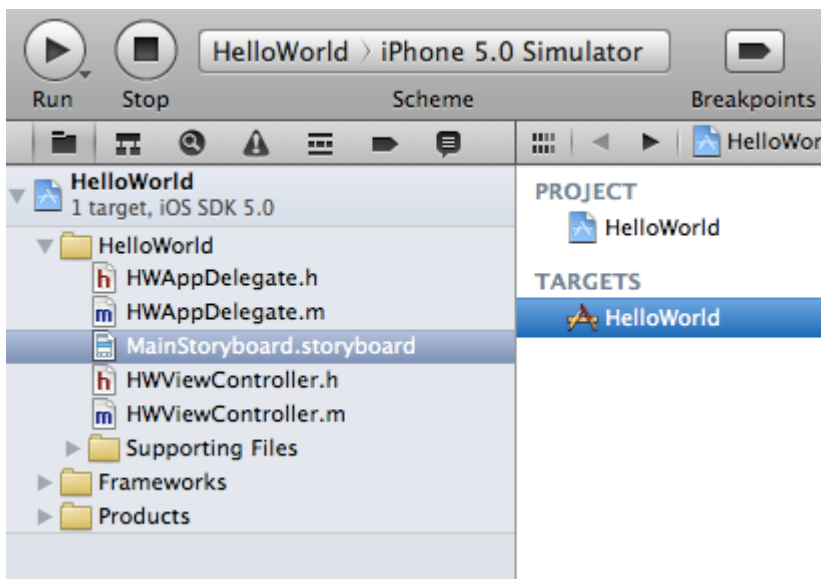


Figure 1.13 Xcode Scheme Selected as iPhone Simulator

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

Click on the Storyboard file in your project (e.g., MainStoryboard.storyboard) and the UI contents are displayed in the Editor which, in this case, is Interface Builder for the user interface (see figure 1.13). That white rectangle is the view controller you saw displayed when you ran the project in the simulator.

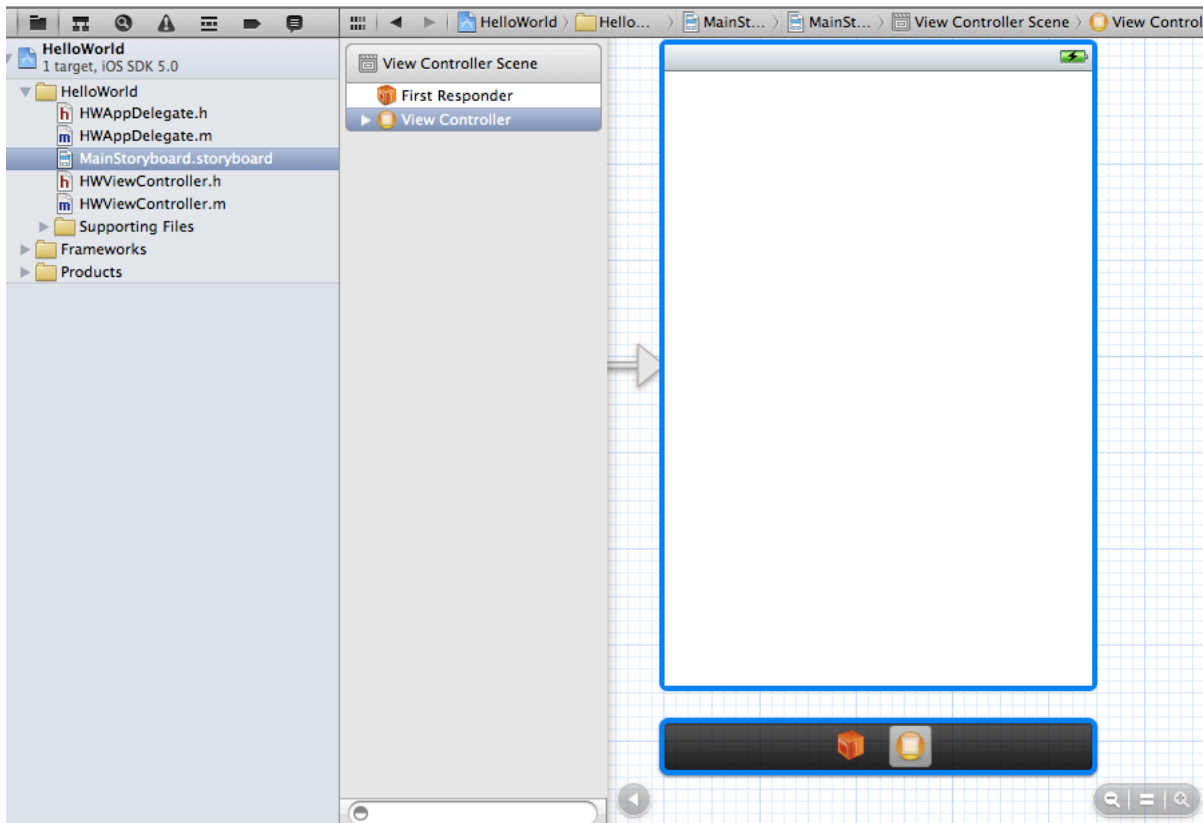


Figure 1.14 Storyboard File Displayed in the Editor

Notice the `HWViewController.h/.m` files in the Navigator. Notice the items in the list on the left in the Editor (see figure 1.13). The one listed as "View Controller" is an instance of our `HWViewController` class. Therefore, changes to it in the Editor will affect how our app runs.

Be sure that Utilities is visible (right button above View on the top right of Xcode) and notice at the bottom there's a list of items (see figure 1.14).

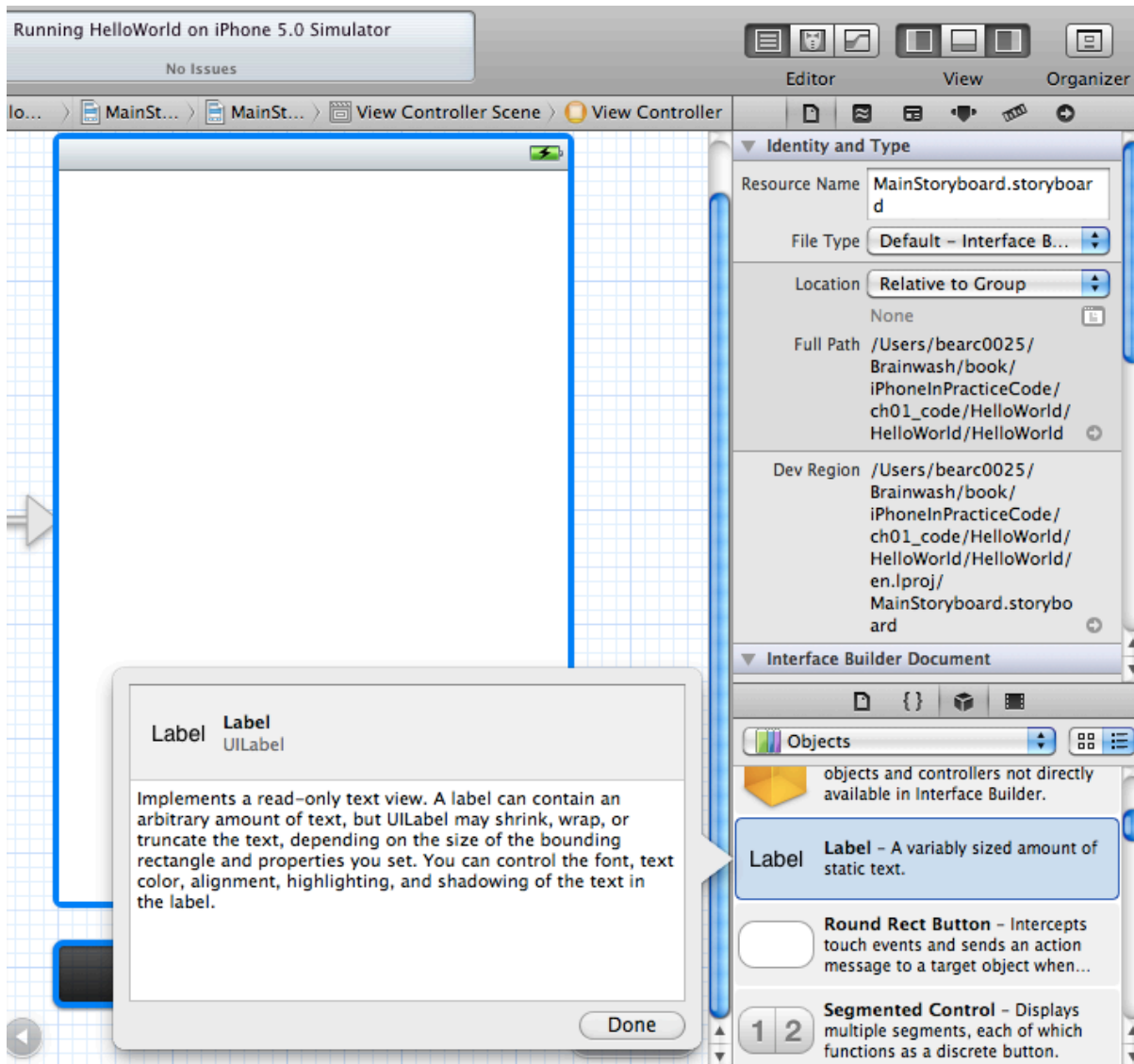


Figure 1.15 Utilities displayed for Storyboard File

That list on the bottom right contains other UI items that can be dropped on to 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.

Double click on the newly dropped Label and type "Hello World" (see figure 1.15). Now run your app again and... congrats.



Figure 1.16 Adding a Label to the Project UI

Now run your app again and... congrats (see figure 1.16).

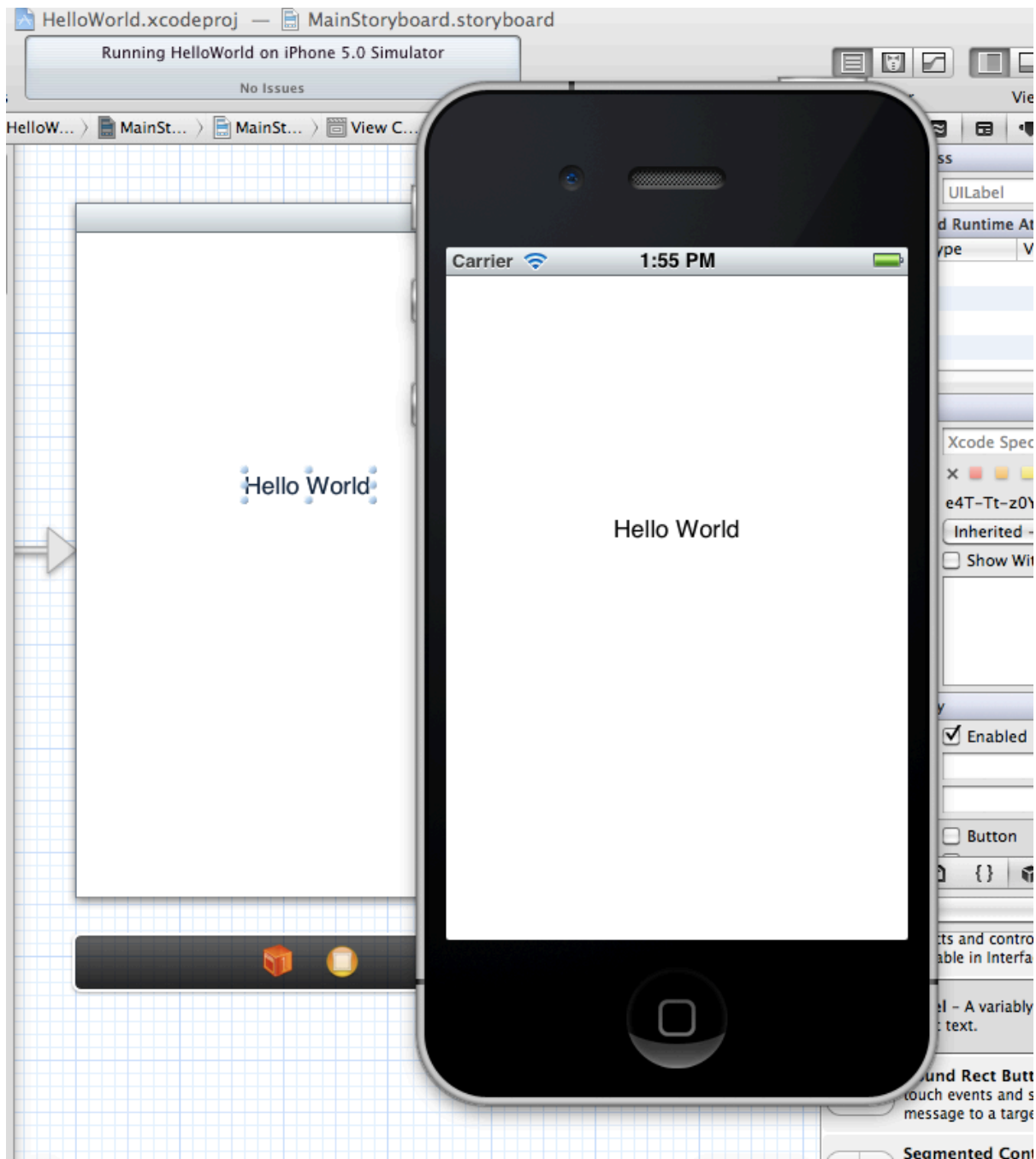


Figure 1.17

1.4 Summary

Great job! Your first app is done! Not so painful, right? Now you're an iOS developer. Though I don't expect you to make much money off of that app.

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

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

Index Terms

- Apple Human Interface Guidelines (HIG)
- Debug Console (Xcode)
- Editor (Xcode)
- Interface Builder
- iOS Developer Program
- Label
- MapKit
- Model-View-Controller (See MVC)
- MVC
- Navigator (Xcode)
- Navigator (Xcode)
- Objective-C (object oriented)
- Organizer (Xcode)
- Simulator
- Single View Application
- Storyboard
- Storyboard
- Storyboard
- Target
- Toolbar (Xcode)
- Utilities (Xcode)
- Utilities (Xcode)
- View Controller
- Xcode
- Xcode
- Xcode
- XIB