iOS 5 SDK apps development using Xcode, Interface Builder, Instruments, GDB, and key tools

Pro iOS 5 Tools

Xcode, Instruments, and Build Tools

Brandon Alexander | J.Bradford Dillon | Kevin Y. Kim

Apress®

Pro iOS5 Tools

Xcode Instruments and Build Tools

Brandon Alexander
Brad Dillion
Kevin Y. Kim

Apress®

Pro iOS5 Tools: Xcode Instruments and Build Tools

ISBN-13 (pbk): 978-1-4302-3608-5

ISBN-13 (electronic): 978-1-4302-3609-2

President and Publisher: Paul Manning
Lead Editor: Steve Anglin and Michelle Lowman
Technical Reviewer: Anselm Bradford
Editorial Board: Steve Anglin, Mark Beckner, Ewan Buckingham, Gary Cornell, Morgan Ertel, Jonathan Gennick, Jonathan Hassell, Robert Hutchinson, Michelle Lowman, James Markham, Matthew Moodie, Jeff Olson, Jeffrey Pepper, Douglas Pundick, Ben Renow-Clarke, Dominic Shakeshaft, Gwenan Spearing, Matt Wade, Tom Welsh
Coordinating Editor: Anita Castro
Copy Editor: Heather Lang and Mary Behr
Compositor: MacPS, LLC
Indexer: SPI Global
Artist: SPI Global
Cover Designer: Anna Ishchenko

Distributed to the book trade worldwide by Springer Science+Business Media, LLC., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail `orders-ny@springer-sbm.com`, or visit `www.springeronline.com`.

For information on translations, please e-mail `rights@apress.com`, or visit `www.apress.com`.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at `www.apress.com/bulk-sales`.

Any source code or other supplementary materials referenced by the author in this text is available to readers at `www.apress.com`. For detailed information about how to locate your book’s source code, go to `http://www.apress.com/source-code/`.

To Erin and Sage, the most beautiful girls in my life. Lub.

—Brandon Alexander

To my wife Jennifer, and our kids, Nevaeh and Jack.
—J. Bradford Dillon

To Annie, for all her love and support.
—Kevin Y. Kim

Contents at a Glance

Contents

About the Authors

Brandon Alexander is an iOS developer, writer and conference speaker living in Decatur, Georgia with his wife and daughter. Brandon began iOS development in 2008 and has spoken at several conferences. As a developer, Brandon enjoys making technology transparent to users and elegantly useful; as a speaker and writer, he loves teaching others about technology so that they can do the same. When he's not coding, writing or speaking at conferences, Brandon enjoys magic and photography. He can be found on twitter at `@whilethis` and via email at `brandon.alexander@gmail.com`.

Coming from an autodidactic background, **Brad Dillon** taught himself the technologies of his trade, working his way through web development, animation, and other desktop-based interactive platforms, finally finding his home in touchscreen interface development. Since the iPhone SDK launched, Brad has been building apps for iOS devices full time. With a passion for user experience and a deep understanding of the technologies at play, Brad has helped to build solutions that bridge the gap between the devices, users, and global brands. He can be found online at `http://jbradforddillon.com`, and followed on Twitter at `@jbradforddillon`.

Kevin Y. Kim is a founder and partner of AppOrchard LLC, a Tipping Point Partners company focused on sustainable iOS development. A graduate of Carnegie Mellon University, he was first exposed to the NeXTStep computer (the ancestor of today's iPhone) while a programmer at the Pittsburgh Supercomputing Center and has been hooked ever since. His career has spanned finance, government, biotech and technology, including Apple where he managed the Apple Enterprise Services team for the New York metro area. He resides in the Alphabet City section of New York City with his wife and a clowder of rescued cats.

About the Technical Reviewer

Anselm Bradford is a lecturer in digital media at the Auckland University of Technology (AUT) in New Zealand, where he researches interactive media, web media, and visual communication. His experience with Internet-related development stretches back to 1996, when he hand-coded his first web site. He may be found `@anselmbradford` on Twitter and occasionally blogs at `AnselmBradford.com`.

Acknowledgments

They say writing a book is a lot of work. This book wasn't any different. Documentation was read and many dead ends were found when looking for the best way to do something. But the end product is something we are all proud of. This book isn't just the product of the three of us; there were many people involved.

The first person I want to thank is Steve Anglin who approached me at 360|iDev and simply asked me if I wanted to write a book. Without knowing how much effort was involved, I reached out to Dave Mark who gave me some sound advice and introduced me to Scott Penberthy who wrote the GDB section and authored our back end.

The next person who deserves a great deal of credit is Anita Castro. Anita kept us in line and pushed us to get all the needed components done and as close to the schedule as possible.

The next set of thank yous goes to our wonderful editors: Ralph Moore, who made sure the text made plenty of sense from a developer's perspective, Anselm Bradford, who followed all of the instructions in the book and made sure everything worked as we explained them, Heather Lang and Mary Behr who helped us all look like great authors, Christine Ricketts who was responsible for the final pieces and production on this book. Without these people, this book would still exist in some files sitting on a server somewhere.

I also want to thank my coauthors, Brad and Kevin. They really stepped up when I was in a pinch and needed help getting this book over the final hump to get it to completion. I know they sacrificed a great deal during the process. Writing a book on a set of tools while they are in beta is not an easy task, so thank you guys!

I also want to thank my wife, Erin, who endured through many hours of me in my office working on this project while waiting on our wonderful daughter, Sage, to enter the world. She kept me focused when I was having writer's block. And I want to thank my wonderful daughter Sage for giving me the motivation to see this project to completion. I couldn't have finished this without the two of you!

Brandon Alexander

Thanks to all of the misfits who publish ideas and share experiences, be it in 140 characters, or hundreds of pages. Thanks to my parents, for teaching me how to teach myself. Most of all, thank you to my wife Jennifer for always supporting me and my work, however geeky she may find it, and to our kids Nevaeh and Jack, for keeping the hours between code sprints both fun and challenging.

J. Bradford Dillon

Thanks to my friends and colleagues at AppOrchard for their patience through the last several months of curmudgeonly behavior and for helping me make this project successful. Thanks to my wife, Annie, for making sure I worked on this book when I would have rather been watching baseball. Thanks to my cats, PK, Manny, Leela and Kit-en, for always walking across the keyboard when I needed a break. KTHXBAI.

Kevin Y. Kim

Introduction

The iOS platform has exploded in popularity over the past few years and is showing no signs of slowing down. The app marketplace has become highly competitive and the users are becoming accustomed to great experiences. This makes our jobs as app developers very difficult. What sets a great application apart from other applications? We're going to attempt to answer that question through the course of this book.

Why Write a Book on Tools?

They say, "An artisan is only as good as his/her tools." This is generally true for most professions. As developers, we usually only need a text editor and a compiler. While that setup can get the job done, a great set of tools starting with the IDE (Integrated Development Environment) and moving to performance analysis and debugging tools can greatly improve a developer's efficiency during the development and debugging process. The main problem a developer faces, especially with new versions of these tools being released, is how does one effectively use these tools? This is the space where this book fits. This book is all about using the amazing developer tools that Apple has provided and how to use them to make our apps great.

How This Book Is Organized

This book is all about process. As you'll read in Chapter 1, no matter what stage you are in, this book will help you on your journey to being a better developer. The first few chapters of this book are all about debugging and performance tuning. We're going to take a project from a state that needs a lot of work to ready for beta testing. Then we take a look at how to improve our efficiency as developers and let the tools do most of our work for us. We will even automate a lot of the process to let us focus on more important issues. The final part of the book is how to share our application with testers, respond to feedback and finally start sharing code between our applications. The last chapter is all about navigating and customizing Xcode to fit our own workflow.

Support and Contributions

If you run into any issues or find a great tip to help those reading this book, head over to `http://proiostools.com/forum/` and participate in the discussion! The goal here is to get a great community going that will intersect with the great iOS development community that already exists. We're here to help and would love to see what kind of tips and tricks you discover while going through the book.

Chapter 1

Wax On, Wax Off

By now, you have written an iOS application or two. You have also learned that making a great app is hard work. From spontaneous crashes to memory leaks and bugs that create other bugs, the simplest of apps can quickly become a nightmare. Fortunately, these issues are easy to diagnose with the tools at our disposal.

That is what this book is mostly about. We have a toolbox available to us as iOS developers. Ultimately, the question quickly becomes: Which tool is best for the task at hand? This book will answer that question for most of your cases. For the cases where there is no obvious answer, you will be equipped with some approaches and techniques that will point you in the right direction.

Who Is This Book For?

In most crafts, the transition from being a complete beginner to being capable is usually swift. The goal during this transition is simply to become functional with the tools and understand the language the craftsmen speak. This transition also builds confidence in the new practitioner. At the end of this transition, practitioners are fully capable of accomplishing most tasks and solving most problems thrown at them.

Most stop at proficient, however. As a magician, I progressed from a newcomer to the art to a proficient amateur relatively quickly. As I learned a new technique or a new effect, I was very excited to practice. As my skill improved, that desire to practice lessened, and I even became bored with rehearsing the same effect or technique over and over. My skill had reached a plateau. I know what I must do to get to the next level, but I don't want to go there right now. That is OK with me.

How is this relevant to software development? First, like any craft, a certain set of programming skills is rapidly acquired, including learning the syntax of a language, understanding flow control, using basic software design patterns, and debugging by writing to standard out and basic use of the provided debugger. The next phase of a developer's path to mastery is learning more about how a language and platform work, more design patterns and their appropriate uses, and more about the debugger. The final, never-ending, phase is simply fine-tuning all of these skills and finding better solutions to existing problems, as well as learning how to reuse code more. This learning path is not the same for every developer. Sometimes, different parts of development are easier to grasp than others.

The point here is that I don't want you to get frustrated when you don't progress as fast as you'd like. Software development is hard. The thing that separates a hobbyist from a professional is the level of commitment. The commitment to go from a proficient hobbyist to a professional generally takes you into career mode. This level of dedication takes more than just hours of practice. You have to start looking at how other software is made. Surrounding yourself with others that develop for the same platform, especially those who are better than you, is key to growing. By purchasing this book, you are also acknowledging that you want to be a better developer. My goal is to help you learn at least one new skill. If you do that, my job is done, and if you do more, even better!

So who is this book for? This book is for those who are ready to reach the next level. Whatever skill level you are at, this book has something for you. Perhaps you are a master at object-oriented programming, and performance tuning is something you want to learn. Or maybe you want to know the best way to create a universal application for iOS without rewriting half your application. The only prerequisite is that you have some exposure to iOS development and Objective-C.

What This Book Is

This book is a guide that will take you from an alpha quality application to a feature-complete and tested application ready for submission to the App Store. This book contains many tricks of the trade, from diagnosing memory issues to tweaking scroll views to squeeze the last bit of performance out of the device. In the end, you'll want to have this book on your desk with pages marked for quick reference on how to solve common problems.

Will this book solve all of your problems? Probably not, but you will walk away with some techniques for solving problems in a very systematic way. Deep down, we're scientists, and following the scientific method for solving problems in software will, in the end, help us learn how to prevent the problem next time.

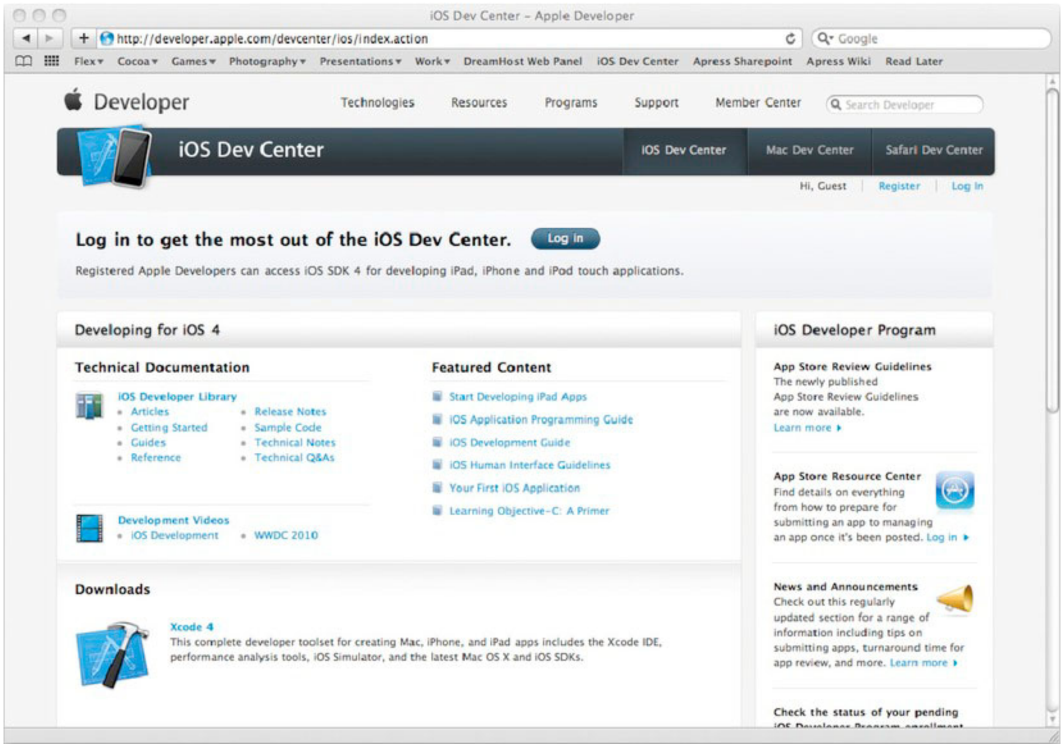
This book also follows a realistic software life cycle. We'll pick up a project at the end of development, and we'll take it through beta testing and finish with a shippable product. We'll hit some common roadblocks and look at how iOS works; we'll even work around some interesting issues. We'll also find some useful libraries written by people who cared enough to share their solutions to particular problems.

What You Need to Get Started

To get the most out of this book, you'll need a paid developer account in the iOS Dev Center. This will give you the ability to test on an iOS device as well as run the performance tools against the iOS device. We will do several things on the device itself, and your best bet is to go ahead and sign up for the paid developer account if you don't have one already. At the time of this writing, the cost is $99 USD for a one-year subscription to the iOS developer program.

If creating an account is not an option for you, you can download Xcode 4 from the Mac App Store. This will give you the Xcode IDE as well as Instruments. You'll have the ability to develop and debug your applications in the iOS Simulator, and you can run some of the performance tools against that simulator. However, you won't be able to do some of the debugging and performance testing that we'll cover later on in this book.

To get set up with the iOS Dev Center, go to `http://developer.apple.com/ios/`. Figure 1–1 shows what the home page looks like. Click the Register link at the top to get started. If you want to deploy your applications to iOS devices, this is your best bet. If you are an enterprise developer, check out the enterprise developer program.



Log in to get the most out of the iOS Dev Center.

Registered Apple Developers can access iOS SDK 4 for developing iPad, iPhone and iPod touch applications.

This complete developer toolset for creating Mac, iPhone, and iPad apps includes the Xcode IDE, performance analysis tools, iOS Simulator, and the latest Mac OS X and iOS SDKs.

The newly published App Store Review Guidelines are now available.

Find details on everything from how to prepare for submitting an app to managing an app once it's been posted.

Check out this regularly updated section for a range of information including tips on submitting apps, turnaround time for app review, and more.

Check the status of your pending

Figure 1–1. *The iOS Dev Center home page*

After you download and install the latest development tools, fire up Xcode, and take a look around. If you are used to Xcode 3, you'll notice Xcode 4 looks completely different. Don't worry; we'll step through the features as we make it through the chapters in this book. If you're new to the platform, say hello to Xcode. I'm sure you'll be friends in no time (don't worry; friends have fights from time to time). Either way, we're in for quite the journey as you learn how to get the most out of the toolbox Apple provides, as well as some great third-party tools.

What's in This Book

This book follows one single project. We're going to pick up this project from an initial alpha state and prepare it for beta testing. Then, we'll walk through the beta testing phase and get some useful feedback from the testers in the form of bug reports and feature requests. You'll learn how to optimize your workflow with automation and migrate to a universal application. The final part is figuring out a great way to share some code with the developer community and finally talk about getting the most out of Xcode 4 and some other useful tools.

Here is a quick overview of each chapter:

- *Chapter 2, First Class Tools (Xcode, Interface Builder and Instruments)*: In this chapter, you'll meet Xcode 4, and the other tools included with the developer tools package for iOS development. Here, we'll talk about new layout of Xcode and what the integration of Interface Builder gives us. We'll also open Instruments, and you'll get acquainted with a tool we'll be using for some of our performance testing and debugging.
- *Chapter 3, Three Screens and. . .Well, It Runs*: Here, we'll dive right into checking out an existing project and take a look at how to use Git and GitHub directly from Xcode. We'll also walk through the first build of our project for this book: Super Checkout.
- *Chapter 4, Memory Management and Diagnostics*: In this chapter, we'll diagnose and solve the number one reason for application crashes. We'll talk about memory management in Objective-C and some best practices. We'll also dive into our first use of Instruments to help diagnose some of these pesky memory issues.
- *Chapter 5, Core Animation and Smooth Scrolling*: Now that we've fixed the memory issues, we'll dive into tuning Core Animation and make our tables scroll like butter. You'll learn your second instrument and some interesting quirks about the rendering model on iOS.
- *Chapter 6, Networking, Cache, and Power Management*: In this chapter, you'll learn all about networking and how the built-in iOS networking APIs work. We'll take a look at a popular networking library and replace the existing networking layer to enhance the application. We'll also talk about caches; we'll explore some caching techniques and why caching might or might not be a good idea. The final part of this chapter will be all about power management. You'll learn about the different radios on an iOS device and how they affect battery drain and how to detect problems.

- *Chapter 7, Prepare the Beta!*: Now that we've addressed some of the big problems in Super Checkout, we'll prepare the application for beta testing. We'll take a look at some beta distribution techniques and some ideas for managing beta testing.
- *Chapter 8, Why are things Breaking?*: We're getting bug reports, the server API changes and all sorts of things are going wrong. In this chapter, we'll take a look at a way to break down our application into testable components to reduce the number of bugs in our code. You'll also learn how to have Instruments automatically drive the application and report any errors. We'll end with a suite of tests to ensure our application is stable and stays that way if we need to make changes.
- *Chapter 9, Can we Automate Some of This?*: We're one step away from a fully automated build system that runs our tests for each new push to source control. We'll meet our trusty build management tool of choice and how to push new builds out to testers automatically.
- *Chapter 10, Now They Want an iPad Version*: Now that feature requests are slowing down, we find that we need an iPad-compatible version of Super Checkout. In this chapter, you'll learn how to migrate Super Checkout to a universal binary and the different techniques for sharing code among the different sizes of iOS devices.
- *Chapter 11, How do I Share Some of This?*: Our application is ready to ship, but we've created some great stuff that we want to break out into a static library so we can share the code among multiple projects and even with the development community as a whole. We'll take a look at how to share the code on GitHub and have a brief discussion over open source licenses.
- *Chapter 12, One More Thing*: By now, we've taken an application from being crash prone and buggy to being something that is ready to ship. Here, we'll look at some other pieces of Xcode and how to speed up our workflow. We'll also look at some great third-party tools that will speed up development and reduce the amount of boilerplate code you have to write.

Here We Go!

Are you ready to get started? Good! A quick note before you turn the page and dive into the project. We're going to cover a lot of information. Trying to take in all of this information in one sitting is probably not a good idea. Take it one chapter at a time, and repeat sections as necessary. Trust me; some of these topics had to be revisited several times before they were put down in this book.

If you get stuck or find yourself getting sleepy, step away for a moment or even grab some shut-eye. This will help you clear your head, and your brain will work on the material while you are doing other things. You'll come back, something will click, and you'll notice that you understand the topic better. Learning is an active process, and as Aaron Hillegass says in *Cocoa Programming for Mac OS X*, "Caffeine is not a substitute for sleep."

Now, fire up Xcode 4, open your notebook, and turn the page to get started. We're going to have some fun.

Chapter 2

First-Class Tools

Xcode has undergone another major revision. This time around, Xcode has one giant window with tabs. A ton of new features with this release make the developer's job easier. Some of these new features include

- A single, unified window that brings everything together.
- The Jump Bar brings quicker navigation through a project as well as a single source file without taking up too much space.
- Interface Builder is fully integrated into Xcode allowing for even tighter integration between the nib and source.
- The Xcode Assistant is a two-pane editor that, when enabled, will pick an appropriate file to view next to the editor in which you are editing.
- LLVM 3.0 is fully integrated into the Xcode which means better syntax highlighting, code completion and many other features that LLVM has.
- Fix-it uses some features of LLVM to not only display compile errors but suggest quick fixes.
- Xcode has better integration with some common version control systems: Git and Subversion (SVN).
- A brand new debugger: LLDB is to GDB as LLVM is to GCC.
- Instruments has a new interface featuring a Jump Bar and other features borrowed from the Xcode interface.

With these new features and enhanced workflows, finding your way around when you are used to Xcode 3 can be a tad frustrating. The plan for this chapter is to take a look at some of the new features of Xcode and see where some of the more common features have moved.

Taking a Look Around

Let's start by creating a simple project and putting some of these things into practice. We're not going to be creating anything meaningful yet; the cool stuff will start in the next chapter. We will be taking a look at where some of the more common tasks can be completed in Xcode 4 as well as some of the new user interface enhancements.

To get started, launch Xcode 4. We're going to create a new project, so click the Create a new Xcode project button shown in Figure 2–1.

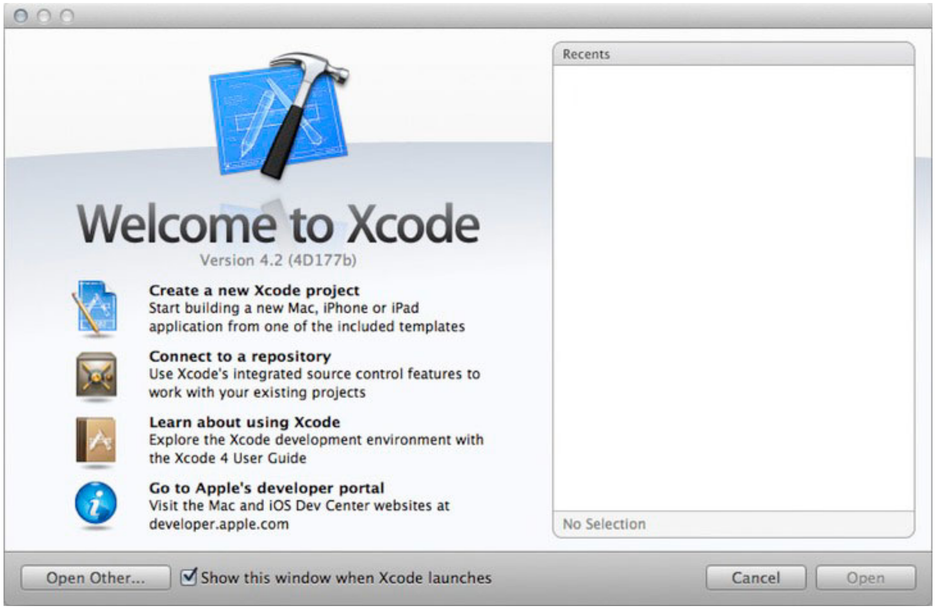
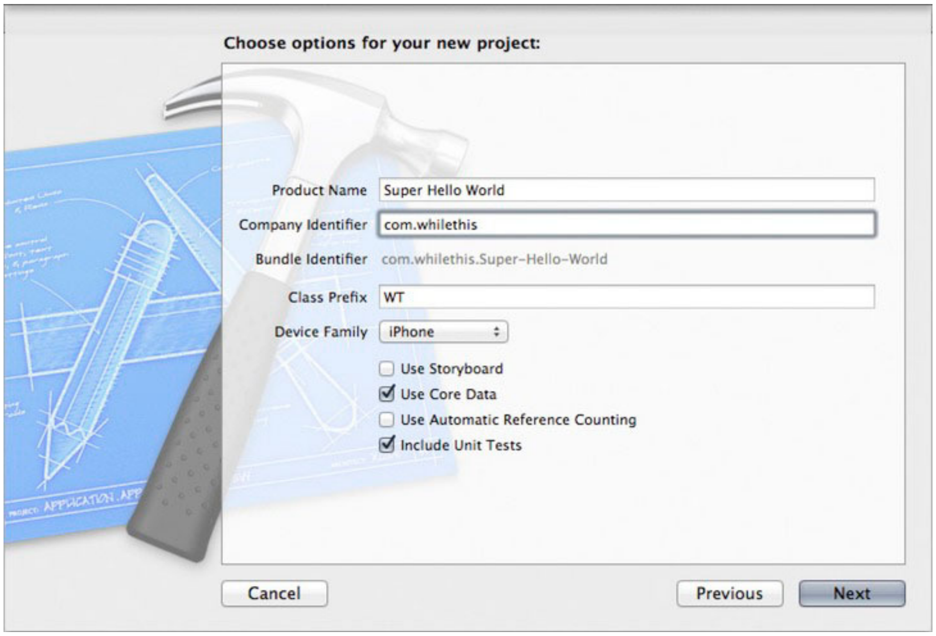

Welcome to Xcode
Version 4.2 (4D177b)
Create a new Xcode project
Start building a new Mac, iPhone or iPad application from one of the included templates
Connect to a repository
Use Xcode's integrated source control features to work with your existing projects
Learn about using Xcode
Explore the Xcode development environment with the Xcode 4 User Guide
Go to Apple's developer portal
Visit the Mac and iOS Dev Center websites at developer.apple.com
Recents
No Selection
Open Other...
Show this window when Xcode launches
Cancel
Open

Figure 2–1. *The familiar screen that greets us when we launch Xcode*

For this chapter, we're going to create a Navigation-based Application. Select that option, and click Next. In the next screen, we name our project, declare our company identifier, and select any other project preferences. Go ahead and name the project **Super Hello World**, and fill in the company identifier in reverse DNS form. For this exercise, we will use **com.example**. Also check Use Core Data and Include Unit Tests for this project. Your screen should look similar to the one shown in Figure 2–2.


Choose options for your new project:

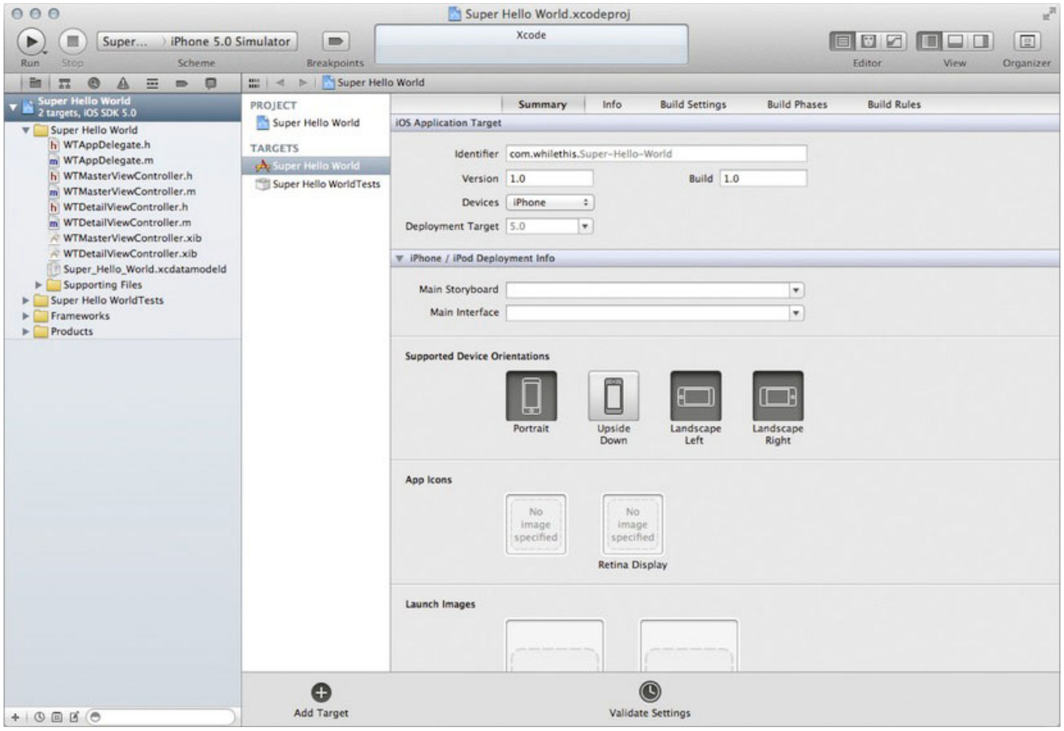
Product Name: Super Hello World
Company Identifier: com.whilethis
Bundle Identifier: com.whilethis.Super-Hello-World
Class Prefix: WT
Device Family: iPhone

- [ ] Use Storyboard
- [x] Use Core Data
- [ ] Use Automatic Reference Counting
- [x] Include Unit Tests

Figure 2–2. *The project naming screen now includes the creation of a unit test target.*

Clicking Next will bring up a sheet asking us where to save the project. Go ahead and choose a location for the project, and select the check box for creating a local Git repository. Clicking Create creates the project and takes you directly into the project settings for Super Hello World.

The first things shown are the project details (see Figure 2–3). The item selected in the middle column is the default target for the project. You will also notice the unit testing target is placed below it. If you want to make any changes to the way the project is compiled and packaged, modify the particular target. If you are used to the way this was done in Xcode 3, you'll notice that modifying targets is now consolidated into this one area; you no longer have to go to the Info pane for modifying the target as you did in Xcode 3.



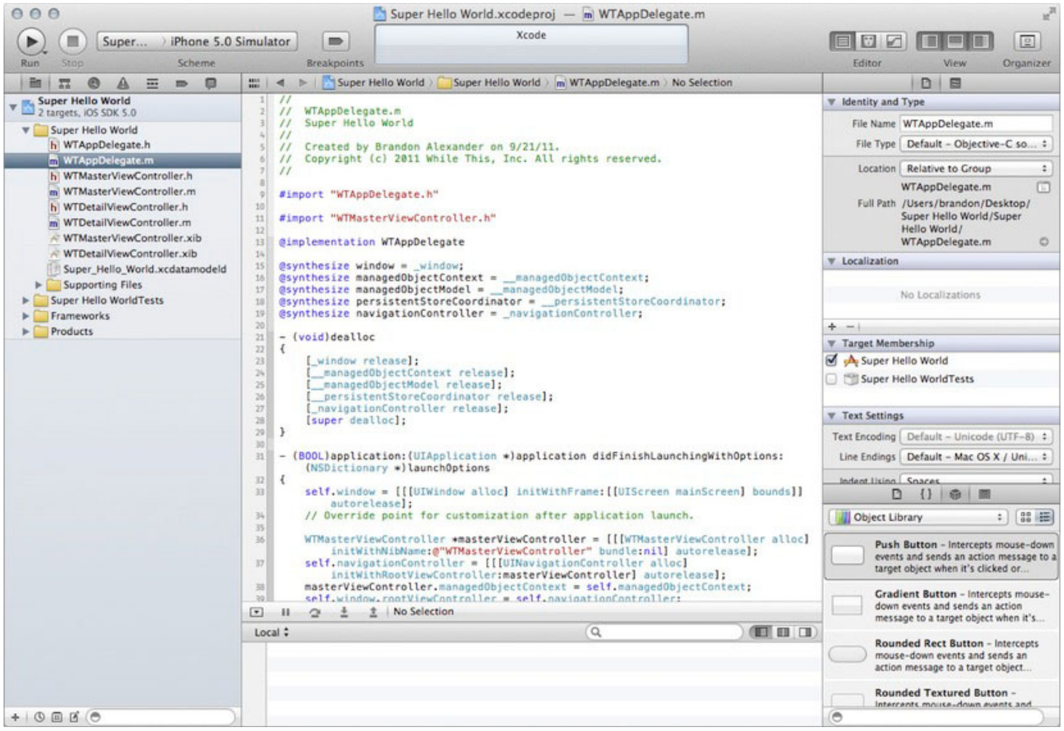
Identifier com.whilethis.Super-Hello-World
Version 1.0 Build 1.0
Devices iPhone
Deployment Target 5.0

Figure 2–3. *Xcode now had a default editor modifying project metadata and targets.*

Now that our project is created, you can see a somewhat familiar view that you are used to in Xcode 3—except there are some new items there.

Figure 2–4 shows the following main sections of the application:

- On the top, the toolbar
- On the left, the Navigator area
- On the bottom, the Debugger area
- On the right, the Utilities area
- In the middle, the Editor area



```
//
//  WTAppDelegate.m
//  Super Hello World
//
//  Created by Brandon Alexander on 9/21/11.
//  Copyright (c) 2011 While This, Inc. All rights reserved.
//

#import "WTAppDelegate.h"

#import "WTMasterViewController.h"

@implementation WTAppDelegate

@synthesize window = _window;
@synthesize managedObjectContext = __managedObjectContext;
@synthesize managedObjectModel = __managedObjectModel;
@synthesize persistentStoreCoordinator = __persistentStoreCoordinator;
@synthesize navigationController = _navigationController;

- (void)dealloc
{
    [_window release];
    [__managedObjectContext release];
    [__managedObjectModel release];
    [__persistentStoreCoordinator release];
    [_navigationController release];
    [super dealloc];
}

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:
    (NSDictionary *)launchOptions
{
    self.window = [[[UIWindow alloc] initWithFrame:[[UIScreen mainScreen] bounds]]
        autorelease];
    // Override point for customization after application launch.

    WTMasterViewController *masterViewController = [[[WTMasterViewController alloc]
        initWithNibName:@"WTMasterViewController" bundle:nil] autorelease];
    self.navigationController = [[[UINavigationController alloc]
        initWithRootViewController:masterViewController] autorelease];
    masterViewController.managedObjectContext = self.managedObjectContext;
    self.window.rootViewController = self.navigationController;
```

File Name: WTAppDelegate.m
File Type: Default – Objective-C so...
Location: Relative to Group
WTAppDelegate.m
Full Path: /Users/brandon/Desktop/Super Hello World/Super Hello World/WTAppDelegate.m

☑ Super Hello World
☐ Super Hello WorldTests

Text Encoding: Default – Unicode (UTF-8)
Line Endings: Default – Mac OS X / Uni...
Indent Using: Spaces

Push Button – Intercepts mouse-down events and sends an action message to a target object when it's clicked or...

Gradient Button – Intercepts mouse-down events and sends an action message to a target object when it's...

Rounded Rect Button – Intercepts mouse-down events and sends an action message to a target object...

Rounded Textured Button – Intercepts mouse-down events and

Figure 2–4. *The Xcode workspace window with every pane open*

So Many Panes!

The first thing you probably noticed is how easy it is to lose precious editor space. The good news is we can close all the panes and focus on code alone. Before we get too far ahead of ourselves, shall we take a look at each section of Xcode and see what it brings us?

At the top is the toolbar. In the toolbar, you can select the active deployment target and run your application. You can also select the appropriate editor for you as well as open and close panes. The button on the far right launches the Organizer window that allows you to manage iOS devices and all of your projects.

The pane on the left is the Navigator area. In this pane, you can choose one of many navigators to interact with. The navigators available to you are Project, Symbol, Search, Issue, Debug, Breakpoint, and Log. The Project navigator is the default navigator that allows you to navigate the files within your project. The Symbol navigator shows you the symbols in the project in a hierarchical or flat view. The Search navigator lets you search your project for text and shows the results below. The Issue navigator shows you any compiler errors or warnings in near-real time with the new compiler technologies built into Xcode.

The next two navigators are for debugging. The Debug navigator is only active during a debug session. You can view information based on threads or queues. Viewing by queues lets you see into the different dispatch queues and shows what kind of queue you're looking at. The Breakpoint pane is where you manage the breakpoints in your application. As with Xcode 3, you can create new breakpoints, move them around, disable them, and remove them.

The final navigator is the Log navigator. Any action you perform that is normally recorded will be placed in here, including builds, static analysis, source control operations, and debug sessions.

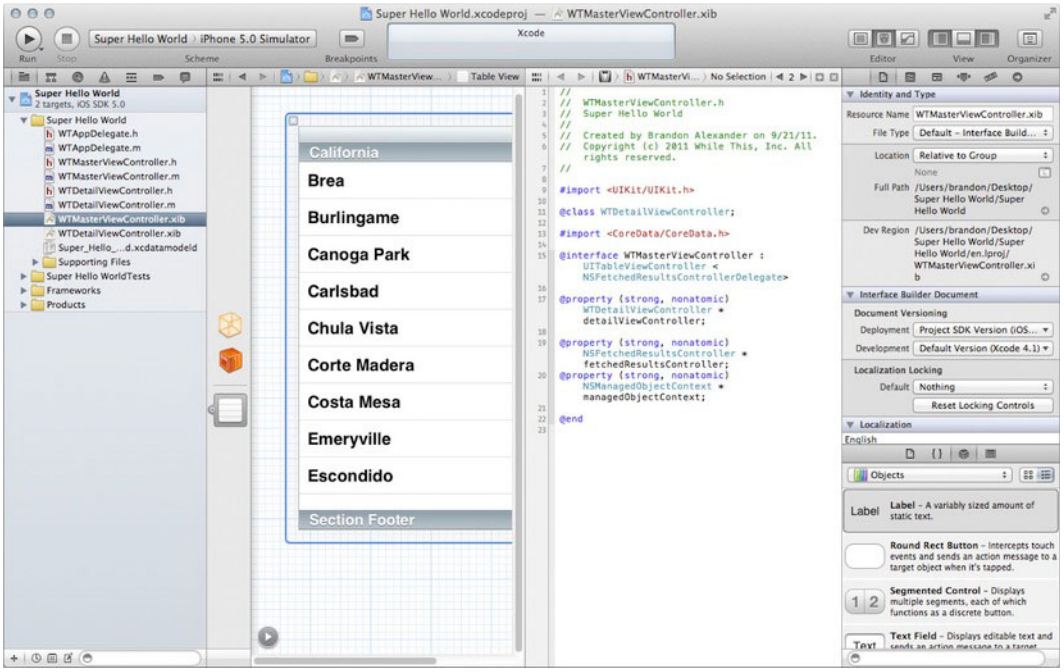
Editors and the Utilities That Follow Them

The area you'll spend most of your time in will be the Editor area. Xcode has several file editors available for you to access:

- Source code
- Project and build settings
- Property list (plist) files
- Rich text files
- Core data models
- Core data mapping models
- XIB (XML nib) files
- AppleScript
- Scripting dictionary files

There are also file viewers that include graphics, videos, and several other file viewers.

There are also three different types of editors to choose from. You can access these editor types from the toolbar. The Standard Editor is your basic editor that lets you edit files as you would normally. The Assistant Editor (see Figure 2–5) is a new type of editor that lets you edit your source files with an assistant showing you a related file next to it. For example, when you are editing a source file and you select the Assistant Editor, the header file is displayed next to it. If you were to open a nib file, the assistant allows you to view the associated header (from the File's Owner property of the nib) and interact with it. From here, you can actually control-drag elements from the nib into the source editor, and Xcode will place the appropriate property in the code where you put it.



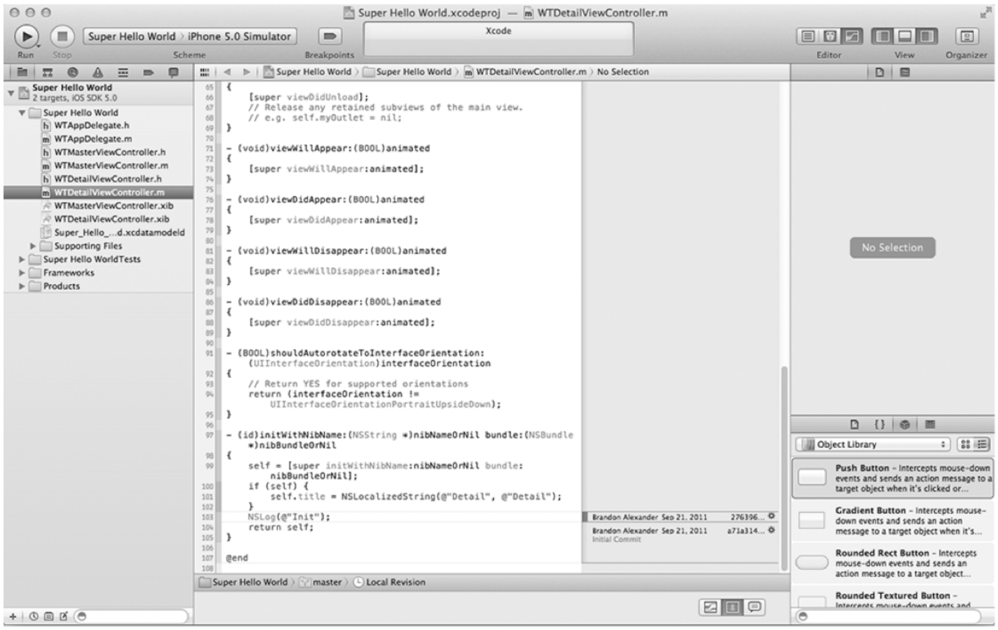
```
//
//  WTMasterViewController.h
//  Super Hello World
//
//  Created by Brandon Alexander on 9/21/11.
//  Copyright (c) 2011 While This, Inc. All
    rights reserved.
//

#import <UIKit/UIKit.h>

@class WTDetailViewController;

#import <CoreData/CoreData.h>

@interface WTMasterViewController :
    UITableViewController <
    NSFetchedResultsControllerDelegate>

@property (strong, nonatomic)
    WTDetailViewController *
    detailViewController;

@property (strong, nonatomic)
    NSFetchedResultsController *
    fetchedResultsController;
@property (strong, nonatomic)
    NSManagedObjectContext *
    managedObjectContext;

@end
```

Super Hello World.xcodeproj — WTMasterViewController.xib
Xcode
Run
Stop
Super Hello World › iPhone 5.0 Simulator
Scheme
Breakpoints
Editor
View
Organizer
Super Hello World
2 targets, iOS SDK 5.0
WTAppDelegate.h
WTAppDelegate.m
WTMasterViewController.h
WTMasterViewController.m
WTDetailViewController.h
WTDetailViewController.m
WTMasterViewController.xib
WTDetailViewController.xib
Super_Hello_...d.xcdatamodeld
Supporting Files
Super Hello WorldTests
Frameworks
Products
California
Brea
Burlingame
Canoga Park
Carlsbad
Chula Vista
Corte Madera
Costa Mesa
Emeryville
Escondido
Section Footer
Identity and Type
Resource Name WTMasterViewController.xib
File Type Default - Interface Build...
Location Relative to Group
Full Path /Users/brandon/Desktop/Super Hello World/Super Hello World
Dev Region /Users/brandon/Desktop/Super Hello World/Super Hello World/en.lproj/WTMasterViewController.xib
Interface Builder Document
Document Versioning
Deployment Project SDK Version (iOS...
Development Default Version (Xcode 4.1)
Localization Locking
Default Nothing
Reset Locking Controls
Localization
English
Objects
Label - A variably sized amount of static text.
Round Rect Button - Intercepts touch events and sends an action message to a target object when it's tapped.
Segmented Control - Displays multiple segments, each of which functions as a discrete button.
Text Field - Displays editable text and sends an action message to a target

Figure 2–5. *The Assistant Editor shows you a nib with the File's Owner header next to it.*

Some file types will not have any counterparts that you can open with the assistant. If this is the case, you can manually select which editor you want next to the selected file. In order to do this, you'll use the Jump Bar in the Assistant Editor to select the file that is shown. You can edit the location of the assistant editor by going to **View ➤ Assistant Editor** and indicating whether the Assistant editor goes on the right or underneath the Standard Editor.

The other editor option is the Version Editor. This editor will show you the changes you've made to the selected file if you have source control turned on. At the time of this writing, Xcode 4 only supports Git and Subversion for source control. From the Version Editor, you can view the history of the file by clicking the time button at the bottom of the space between the diff viewer. From the same screen, you can also view the blame history as well as a log-annotated view of the file's history (see Figure 2–6).
]



```
{
    [super viewDidUnload];
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
}

- (void)viewWillAppear:(BOOL)animated
{
    [super viewWillAppear:animated];
}

- (void)viewDidAppear:(BOOL)animated
{
    [super viewDidAppear:animated];
}

- (void)viewWillDisappear:(BOOL)animated
{
    [super viewWillDisappear:animated];
}

- (void)viewDidDisappear:(BOOL)animated
{
    [super viewDidDisappear:animated];
}

- (BOOL)shouldAutorotateToInterfaceOrientation:
    (UIInterfaceOrientation)interfaceOrientation
{
    // Return YES for supported orientations
    return (interfaceOrientation !=
        UIInterfaceOrientationPortraitUpsideDown);
}

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle
    *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:
        nibBundleOrNil];
    if (self) {
        self.title = NSLocalizedString(@"Detail", @"Detail");
    }
    NSLog(@"Init");
    return self;
}

@end
```

```
Brandon Alexander  Sep 21, 2011   276396...
Brandon Alexander  Sep 21, 2011   a71a314...
Initial Commit
```

Push Button – Intercepts mouse-down events and sends an action message to a target object when it's clicked or...

Gradient Button – Intercepts mouse-down events and sends an action message to a target object when it's...

Rounded Rect Button – Intercepts mouse-down events and sends an action message to a target object...

Rounded Textured Button – Intercepts mouse-down events and

Figure 2–6. *The Version Editor gives you the ability to view the file's history as well as version annotations, affectionately known as the blame view.*

In Figure 2–6, you can also see the Utilities area to the right of the editors. This pane shows contextual information based on your selection. In Figure 2–5, we have the `RootViewController` nib selected, and you can see that the Utilities area looks similar to the Inspector window from Interface Builder in the previous version of Xcode. If you select a source file, you'll see the same information you would see when clicking Get Info in Xcode 3. From there, you can select which target the source file belongs to as well as numerous other tidbits about the file.

As you navigate through the source file, you can click the Show Quick Help button of the Utilities area to see brief information about any symbols the cursor is in. From there, you can click any blue item, which may open the headers file, other documentation, or sample code depending on the context of what was clicked. Take a moment to play around with the different editors and get familiar with the new interface. Don't worry; I'll be here when you get back.

Jump Bars

While you were getting acquainted to the editor, did you notice the bar at the top? It is called the Jump Bar, which allows you to navigate quickly through your file. Figure 2–7 shows the Jump Bar for the `RootViewController` in the project we have created.

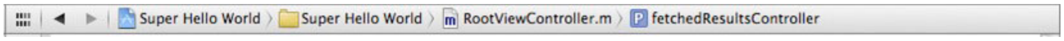


Figure 2–7. *The Jump Bar is a navigation tool that is always available to the editor it is above. It is also available in Instruments to perform the same types of tasks.*

Using the Jump Bar is a quick way to navigate through your project without having to find the file using the Project navigator. The Related Files button (the one that looks like an equal sign) pops up a list that allows you to see recent files, unsaved files, and numerous other lists that pertain to the file currently being viewed.

The back and forward buttons allow you to navigate through your history of files in the editor that is open. Swiping left and right with two fingers will allow you to quickly navigate through the files, as you can within Safari on Lion.

Everything else to the right is a breadcrumb trail showing how to get to the method you are editing, starting with the project. In Figure 2–7, you see we started with the project and in the Super Hello World group is the file `RootViewController.m`, and we are looking at the `fetchedResultsController` selector. Clicking any of the items displays a list of siblings and allows you to drill down into each successive level.

The Organizer

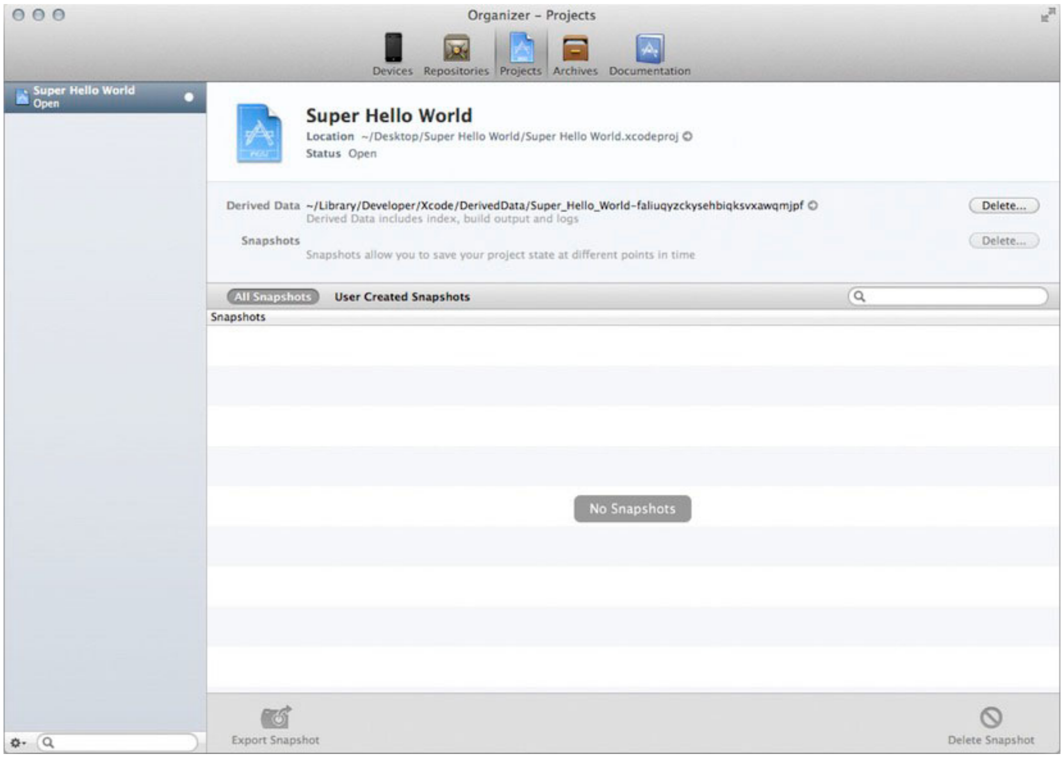
Xcode 4 brings a new organizer that does pretty much what the name implies: it organizes your development workspace. You can open it by clicking the Organizer button on the toolbar or by pressing ⇧⌘2. Figure 2–8 shows the Organizer open to the Projects tab. The Organizer has five main purposes:

- Manage development devices
- Manage source control repositories
- Manage recent projects
- Manage archived builds
- Show documentation

Take a look at the Devices section. If you are a registered developer, you can manage all of your developer profile, provisioning profiles, old iOS software images, device logs, and screenshots from this section. You can also see specific device information that Xcode collects on each device when you plug it in. When you plug in a device, you can take screenshots (for App Store submissions) and even save a screenshot as the launch image.

The next tab to look at is the Repositories tab. It shows the repositories for recent projects that have been set up with source control. You can set up repository links and check out new working copies of repositories if there is an existing project you are checking out. As was mentioned previously, Xcode only supports Git and Subversion. Any other source control will have to be managed from an external tool.

Figure 2–8 shows the Projects tab. This tab shows you the projects that Xcode knows about. It shows you which projects are currently open and other basic information about the project. This view also lets you manage project snapshots.



Super Hello World
Location ~/Desktop/Super Hello World/Super Hello World.xcodeproj
Status Open

Derived Data ~/Library/Developer/Xcode/DerivedData/Super_Hello_World-faliuqyzckysehbiqksvxawqmjpf
Derived Data includes index, build output and logs

Snapshots
Snapshots allow you to save your project state at different points in time

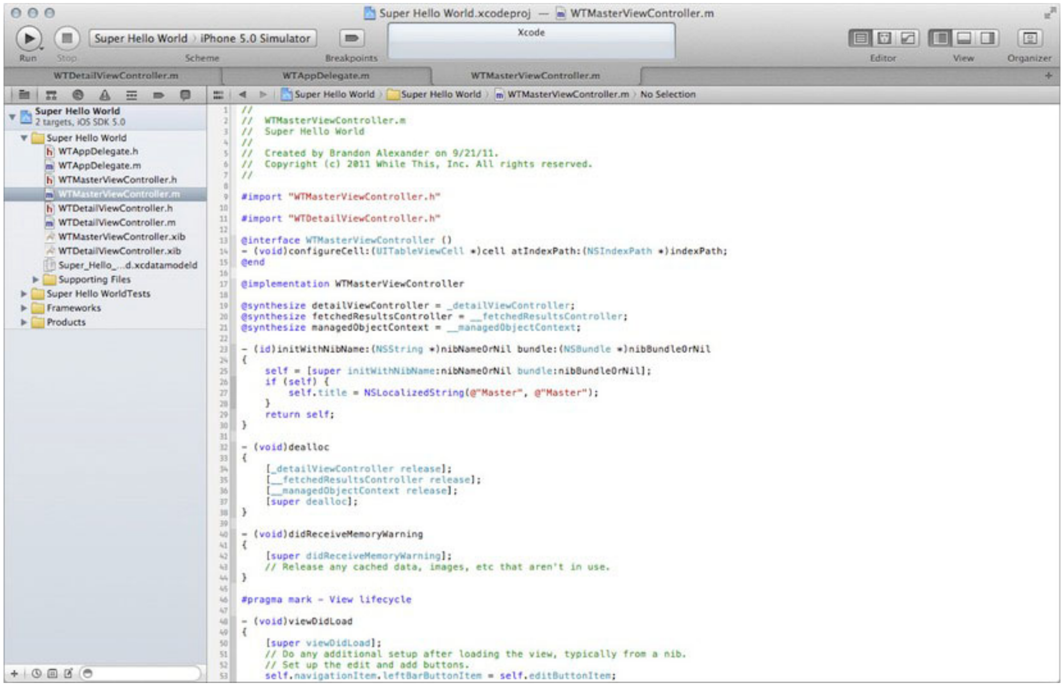
Figure 2–8. *The Organizer with the list of projects you've worked with*

The next tab lets you manage archived applications. Application archiving was brought to Xcode in version 3 to allow iOS developers an easy way to submit applications for approval. After archiving the application, you can validate the binary to check for commonly missed items before you submit the application to the store, share an ad hoc build with beta testers, or submit the application to the store.

The final tab in the organizer is the documentation tab. This is your gateway to the documentation for iOS, Mac OS, and Xcode 4. While looking at any documentation, you'll find a Jump Bar at the top, which acts just like the Jump Bar in your source editors.

Tabs, Tabs, and More Tabs

Let's look at one last item before we head back to our project: tabs. Xcode has tabs that allow you to have multiple editors open at once in the same window. Figure 2–9 shows Xcode with three tabs open.



```
//
//  WTMasterViewController.m
//  Super Hello World
//
//  Created by Brandon Alexander on 9/21/11.
//  Copyright (c) 2011 While This, Inc. All rights reserved.
//

#import "WTMasterViewController.h"

#import "WTDetailViewController.h"

@interface WTMasterViewController ()
- (void)configureCell:(UITableViewCell *)cell atIndexPath:(NSIndexPath *)indexPath;
@end

@implementation WTMasterViewController

@synthesize detailViewController = _detailViewController;
@synthesize fetchedResultsController = __fetchedResultsController;
@synthesize managedObjectContext = __managedObjectContext;

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
        self.title = NSLocalizedString(@"Master", @"Master");
    }
    return self;
}

- (void)dealloc
{
    [_detailViewController release];
    [__fetchedResultsController release];
    [__managedObjectContext release];
    [super dealloc];
}

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

#pragma mark - View lifecycle

- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.
    // Set up the edit and add buttons.
    self.navigationItem.leftBarButtonItem = self.editButtonItem;
```

Figure 2–9. *Multiple tabs within Xcode*

These tabs bring more than just multiple editors. You can pull the tabs out of the window and open a brand new window with that configuration. This means you can have one window open for editing source code and another for editing a nib file. Each tab can be configured and has its own history management. If there is only one tab open, you can show or hide the tab bar by going to **View ➤ Show/Hide Tab bar**.

Why don't you go ahead and navigate through the different views within Xcode? I'll still be here when you get back.

Getting Back to the Code

Now that we've taken a more in-depth look at Xcode, let's do some basic modifications to our project and start checking in changes to our local Git repository. In this chapter, we're going to build a very simple data collection application. We're going to modify the core data model, add a new view controller, and then check all of the changes into source control.

Before we begin, build the application to make sure it compiles. Go to **Product ➤ Build** or press ⌘B. The project should compile with no errors. Clean the project to purge any build artifacts by going to **Product ➤ Clean** or pressing ⌘⇧K.