

Fritz Anderson

**Completely
Revised**

Covers Xcode 4
for iOS and Mac
Development

Xcode 4

UNLEASHED

SAMS

FREE SAMPLE CHAPTER



SHARE WITH OTHERS

Praise for *Xcode 4 Unleashed*

“There are many great resources out there for learning iOS and Mac development that cover Objective-C and Cocoa. Xcode is an extremely important part of iOS and Mac development that often gets overlooked. You owe it to yourself to understand Xcode and all of its quirks and power user features to achieve maximum efficiency as a developer. *Xcode 4 Unleashed* can help you do just that.”

—Tony Hillerson,
Member and Software Architect, Tackmobile.com

“Fritz Anderson’s *Xcode Unleashed* series is the definitive guide to using Xcode. *Xcode 4 Unleashed* has been rewritten to cover the sweeping changes in recent versions of the product. I highly recommend this book to anyone who uses Xcode—newbies and grizzled veterans alike.”

—Duncan Champney,
Director of Software Development, WareTo

Praise for *Xcode 3 Unleashed*

“I would recommend this book to anyone that is serious about programming on the Mac. It is an excellent resource; I plan to refer to it often.”

—Cortis Clark

“I’ve been doing Mac OS X development for seven years, so I was surprised at how much new information I learned in this book. The details on building and the overview of Instruments were invaluable.”

—Dan Wood, Karella Software

“There isn’t a better book on the market to understand Apple’s powerful—yet free integrated development environment, Xcode. Fritz Anderson stands among the most literate programmers I know, simultaneously able to provide a high-level development narrative while delving into the countless crucial details that make up modern development. I recommend *Xcode 3 Unleashed* to both novices as an introduction and professionals as a reference.”

—Jonathan ‘Wolf’ Rentzsch, <http://rentzsch.com>

“Whether you are new to programming on Mac OS X or a seasoned veteran, *Xcode 3 Unleashed* has something for you. The book is full of examples and practical information. I recommend this book for anyone doing serious development on Mac OS X 10.5.”

—Dave Dribin

Fritz Anderson

Xcode[®] 4

UNLEASHED

SAMS

800 East 96th Street, Indianapolis, Indiana 46240 USA

Xcode® 4 Unleashed

Copyright © 2012 by Pearson Education, Inc.

All rights reserved. No part of this book shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without written permission from the publisher. No patent liability is assumed with respect to the use of the information contained herein. Although every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained herein.

ISBN-13: 978-0-672-33327-9

ISBN-10: 0-672-33327-9

The Library of Congress cataloging-in-publication data is on file.

Printed in the United States of America

First Printing May 2012 with corrections November 2012

Trademarks

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Sams Publishing cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Warning and Disclaimer

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an “as is” basis. The author and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book or programs accompanying it.

Bulk Sales

Sams Publishing offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales. For more information, please contact

U.S. Corporate and Government Sales

1-800-382-3419

corpsales@pearsontechgroup.com

For sales outside of the U.S., please contact

International Sales

international@pearson.com

Editor-in-Chief

Mark Taub

Acquisitions Editors

Trina MacDonald
Chuck Toporek

Managing Editor

Kristy Hart

Project Editor

Jovana San Nicolas-
Shirley

Copy Editor

Apostrophe Editing
Services

Indexer

Erika Millen

Proofreaders

Jess DeGabriele
Chrissy White

Technical Editors

Duncan Champney
Tony Hillerson
George Sealy
Rob Wittner

Publishing

Coordinator
Olivia Basegio

Cover Designer

Gary Adair

Senior Compositor

Gloria Schurick

Contents at a Glance

Introduction	1
Part I First Steps	
1 Getting Xcode	9
2 Kicking the Tires	17
3 Simple Workflow and Passive Debugging	25
4 Active Debugging	35
5 Compilation	45
6 Adding a Library Target	57
7 Version Control	65
Part II The Life Cycle of an iOS Application	
8 Starting an iOS Application	87
9 An iOS Application: Model	99
10 An iOS Controller	113
11 Building a New View	127
12 Adding Table Cells	143
13 Unit Testing	155
14 Measurement and Analysis	173
15 Storyboard	197
16 Provisioning	221
Part III Xcode for Mac OS X	
17 Starting a Mac OS X Application	239
18 Wiring a Mac Application with Bindings	253
19 A Custom View for Mac OS X	275
20 Localization and Autolayout	285
21 Bundles and Packages	307
22 Frameworks	325
23 Property Lists	337

Part IV Xcode Tasks

24	Xcode 4 for Xcode 3 Veterans	353
25	Documentation in Xcode	369
26	The Xcode Build System	389
27	Instruments	411
28	Snippets	437

Part V Appendixes

A	Objective-C	455
B	Some Build Variables	473
C	Project and Target Templates	485
D	Resources	499

Table of Contents

Introduction	1
Part I First Steps	
1 Getting Xcode	9
Before You Do Anything	9
Requirements	10
Installing Xcode	10
What You Get	11
Removing Xcode	12
Apple Developer Programs	12
Through an Installer Package	13
Summary	15
2 Kicking the Tires	17
Starting Xcode	17
Hello World	19
A New Project	19
Quieting Xcode Down	21
Building and Running	21
The Real Thing	23
Getting Rid of It	23
Summary	24
3 Simple Workflow and Passive Debugging	25
Building	28
Running	30
Simple Debugging	32
Summary	33
4 Active Debugging	35
A Simple Test Case	35
Going Active	35
Setting a Breakpoint	36
The Variables Pane	37
Stepping Through	38

Fixing the Problem	40
Behaviors	40
The Fix	42
Summary	43
5 Compilation	45
Compiling	46
Linking	50
Dynamic Loading	51
Xcode's Refinements	52
Compiler Products	55
Intermediate Products	55
Precompiled Headers	56
Summary	56
6 Adding a Library Target	57
Adding a Target	57
Targets	58
Target Membership	58
Adding Files to a Target	59
Headers in Targets	61
A Dependent Target	62
Adding a Library	63
Debugging a Dependent Target	63
Summary	64
7 Version Control	65
Taking Control	66
Creating a Git Repository by Hand	66
The State of Your Files	68
How Subversion Views Files	68
How Git Views Files	68
How Xcode Views Files	69
Your First Commit	70
Adding a Remote Repository	71
Setting Up the Remote	71
Pushing to the Remote	72
Starting from a Repository	74
Merges and Conflicts	75
User A	75
User B	75
Merging	76
Conflicts	77

The Versions View	79
Comparison	79
Blame	81
Log	82
Branching	82
Summary	84

Part II The Life Cycle of an iOS Application

8 Starting an iOS Application	87
Planning the App	87
Model-View-Controller	87
The Model	88
The Views	89
The Controllers	90
Starting a New iPhone Project	90
Target Editor	92
Copyright, Again	93
One More Thing	97
Summary	98
9 An iOS Application: Model	99
Implementing the Model	99
Entities	100
Attributes	100
Relationships	102
Managed-Object Classes	105
Creating the Classes	105
Extending the Classes	106
Some Test Data	108
Making the Model Easier to Debug	111
Summary	111
10 An iOS Controller	113
Renaming Symbols	113
Refactoring a Method Name	114
Refactoring a Class Name	114
Editing the View Controller	116
The Table View	116
Setting Up the Passer List	117
Creating a New Passer	117
Live Issues and Fix-it	118

The Real Passer Rating	120
Another Bug	120
Running Passer Rating	123
Summary	125
11 Building a New View	127
Adding a View Controller	127
XIB Files	128
Building a View	130
Lots of Labels	132
First Tryout	134
Outlets	134
Checking Connections	137
Connecting GameController	137
Code Completion and Snippets	139
Testing the Passer Detail View	141
Summary	141
12 Adding Table Cells	143
The Game Table	143
Schemes	147
A Custom Table Cell	149
Summary	154
13 Unit Testing	155
Logic Testing	156
Test Data	158
Testing the CSV Reader	159
Application Testing	166
SenTestingKit Assertions	168
Simple Tests	169
Equality	169
Exceptions	169
Summary	170
14 Measurement and Analysis	173
Speed	173
Memory	182
Allocations	182
Leaks	187
Zombies	189

Analysis	193
The Analyzer	193
Automatic Reference Counting	195
Summary	196
15 Storyboard	197
What Storyboard Is	197
A Storyboard Project	199
Reconstructing Passer Rating	201
Workspaces	201
Copying the Model	203
Coding the Passer List	205
Copying Views	205
A Custom Table View Cell	207
Adding a Passer Editor	210
Creating the Editor View	210
Coding the Editor Controller	212
Adding a Segue	215
Editing an Existing Passer	217
Summary	219
16 Provisioning	221
Developer Programs	221
Organizations	221
Individuals	222
The Enterprise Program	222
The Provisioning Story	222
Automatic Device Provisioning	223
The Provisioning Portal	225
Development Certificates	225
Distribution Certificates	225
Device IDs	226
Application IDs	227
Development Profiles	228
Distribution Profiles	229
Using a Signing Identity	230
Distribution Builds	231
Sharing Identities and Profiles	233
Preparing an App Store Release	234
Final Provisioning	234
iTunes Connect	234
Validating and Submitting	235
Summary	236

Part III Xcode for Mac OS X

17	Starting a Mac OS X Application	239
	The Goal	239
	Getting Started	240
	Model	243
	Porting from iOS	243
	Automatic Reference Counting	246
	Making the Application Twitch	248
	Wiring Up a Menu	248
	Loading Data into LeagueDocument	250
	Summary	251
18	Wiring a Mac Application with Bindings	253
	Filling the Document Window	253
	A Table View	254
	Autosizing	255
	Your First Object Controller	258
	Binding the Team Table	260
	Running Bindings	260
	Laying Out Views	263
	The Passer and Game Array Controllers	264
	Binding the Passer Table	266
	The Game Table—Truncation and Dates	268
	The Game Popover	269
	Summary	273
19	A Custom View for Mac OS X	275
	A Graphing View	276
	Back to the View Controller	279
	Using PasserGraphController	281
	Custom View Properties	282
	Summary	283
20	Localization and Autolayout	285
	Adding a Localization	286
	Trying It Out	287
	Localizing MainMenu.xib	288
	Localizing the Window XIBs	291
	Translating View Strings	291
	Making the Text Fit—by Hand	292
	Making the Text Fit—Autolayout	292

Localizing Info.plist	300
Strings in Code	302
Summary	306
21 Bundles and Packages	307
A Simple Package: RTFD	308
Bundles	309
Application Bundles	309
The Info.plist File	311
Localizing Info.plist	312
Info.plist Keys	312
Keys for All Bundles	312
Keys for iOS and Mac OS X Applications	314
Keys for Mac OS X Applications	315
iOS Keys	320
Keys for Plug-ins	322
Keys for Preference Panes	323
Keys for Dashboard Widgets	323
Summary	324
22 Frameworks	325
Adding a Framework Target	326
Populating the Framework	326
Using the Framework	327
Installing a Framework	327
Running the Application Alone	328
Where Frameworks Are Found	330
Putting the Framework in the Application	331
Building Mac Passer Rating	332
One More Thing	332
Summary	336
23 Property Lists	337
Property List Types	337
Editing Property Lists	338
A Brand New Property List	341
Why Not the Property List Editor?	345
Other Formats	348
Text Property Lists	348
Binary Property Lists	348
Specialized Property Lists	349
Summary	350

Part IV Xcode Tasks

24	Xcode 4 for Xcode 3 Veterans	353
	The Desktop and the Browser	353
	Start Slow	354
	The Sorcerer's Apprentice	355
	The Editor	355
	The Assistant Editor	355
	More Than One Editor	356
	Building	358
	Where Did Everything Go?	358
	Groups & Files	358
	Detail View	360
	Info Windows	360
	Special-Purpose Editors	362
	Browsers	364
	Source Control	364
	Interface Builder	365
	Other Changes	366
	Summary	368
25	Documentation in Xcode	369
	Intrinsic Help	369
	The Quick Help Inspector	369
	The Quick Help Popover	370
	Open Quickly	371
	Help	372
	Xcode How-To's	373
	The Documentation Organizer	373
	Browsing Documentation	373
	Searching Documentation	374
	Bookmarks	375
	Keeping Current	375
	Generating Documentation	377
	Installing Doxygen	378
	What Doxygen Does	378
	Configuring Doxygen: The Wizard	381
	Configuring Doxygen: Expert Settings	383
	Running Doxygen	384
	Installing a Docset	385
	Making Doxygen Part of Your Builds	386
	Summary	388

26	The Xcode Build System	389
	Xcode Build Variables	392
	Settings Hierarchy	393
	Editing Build Variables	395
	Configurations	396
	Adjusting Configurations	396
	Adding Configurations	398
	Configuration Files	398
	Creating a Configuration File	398
	SDK- and Architecture-Specific Settings	399
	Preprocessing xcconfig Files	399
	The <code>xcodebuild</code> Tool	400
	Custom Build Rules	401
	The Build Log	403
	A Simple Build Transcript	404
	Resources	406
	Precompiled Header	407
	Compiling Source Files	408
	Linking	409
	Making a Universal Binary	410
	Touch	410
	Summary	410
27	Instruments	411
	What Instruments Is	411
	Running Instruments	412
	The Trace Document Window	413
	The Library	419
	Instrument Configuration	420
	Recording	421
	Saving and Reopening	423
	The Instruments	424
	Core Data	424
	Custom Instruments	425
	Dispatch	425
	File System	425
	Garbage Collection	426
	Graphics	426
	Input/Output	426
	Master Tracks	426
	Memory	426
	System	427

Threads/Locks	429
UI Automation	429
User Interface	430
Instruments Available to iOS	430
Custom Instruments	431
The Templates	433
For Both Mac and iOS	433
iOS Only	434
Mac Only	435
Summary	435
28 Snippets	437
Tricks	437
General	437
The Jump Bar	440
Code Folding Ribbon	440
The Assistant Editor	441
Interface Builder	442
Instruments and Debugging	443
Building	445
Managing Schemes	447
Traps	448
Part V Appendixes	
A Objective-C	455
The Basics	456
A Class Interface	457
A Class Implementation	458
Objective-C 2.0 and Cocoa	460
Key-Value Coding	461
Memory Management	462
Attribute Accessors and Memory Management	463
Properties	464
Fast Enumeration	467
Foundation Data Types	468
Dynamic Dispatch	470
Objective-C++	471
Summary	471

B	Some Build Variables	473
	Useful Build Variables	475
	Environment	475
	Build Targets	477
	Source Locations	478
	Destination Locations	478
	Bundle Locations	479
	Compiler Settings	480
	Search Paths	481
	Deployment	482
	Info.plist	482
	Source Trees	483
C	Project and Target Templates	485
	iOS Project Templates	487
	Application	487
	Framework & Library	488
	Other	488
	Mac OS X Project Templates	489
	Application	489
	Framework & Library	490
	Application Plug-in	491
	System Plug-in	492
	Other	492
	Target Templates	493
	iOS File Templates	493
	Cocoa Touch	493
	C and C++	494
	Core Data	495
	Resource	495
	Other	496
	Mac OS X File Templates	496
	Cocoa	496
	C and C++	496
	User Interface	497
	Core Data	497
	Resource	497
	Other	497
	The File Template Library	497
	Summary	498

D Resources	499
Books	499
On the Net	500
Forums	500
Mailing Lists	501
Developer Technical Support	501
Sites and Blogs	502
Face-to-Face	503
Meetings	503
Classes	503
Other Software	504
Text Editors	504
Accessories	505
Index	507

About the Author

Fritz Anderson has been writing software, books, and articles for Apple platforms since 1984. He has worked for research and development firms, consulting practices, and freelance. He was admitted to the Indiana bar, but thought better of it. He is now an iOS and Mac programmer for the Scholarly Technology department at the University of Chicago. He has two daughters.

Dedication

*For Steve Jobs, who made my life's work possible,
and
for Kate and Bess, who made my life.*

Acknowledgments

Only part of the effort that went into putting *Xcode 4 Unleashed* into your hands was spent at a text editor. I owe a debt of thanks to those without whom this book could not have been made.

Chuck Toporek, my editor, showed patience, good humor, and good sense in support of our second effort together. *Xcode 3.2 Unleashed* had reached nearly 400 pages when Apple announced Xcode 4, which would not see its first release for 8 months; major revisions spanned another 8. He encouraged a dispirited author through the long process of dumping everything, waiting, and starting again.

Chuck's assistant, Olivia Basegio, made sure the contracts, correspondence, (and advance payments!) all got through.

Trina MacDonald took over after Chuck left for Apple. She, and Jovana San Nicolas-Shirley, who oversaw production, were extremely accommodating with an author whose opinions on how to write a book are... exacting.

I'm especially grateful for the advice of the technical reviewers Chuck found for me. You did yeoman service in the face of a subject in rapid flux and saved me many embarrassments. Of course, any errors that remain are my own.

Quinn Dombrowski, boss and friend, took a kind interest in my book. She was generous with vacation time so I could get it done and patient when I began to fray under the strain of 60-hour weeks.

Bess and Kate bore more than daughters should of my doubts and frustrations, and were simply confident that I would do fine—which was all they needed to do.

We Want to Hear from You!

As the reader of this book, *you* are our most important critic and commentator. We value your opinion and want to know what we're doing right, what we could do better, what areas you'd like to see us publish in, and any other words of wisdom you're willing to pass our way.

You can email or write me directly to let me know what you did or didn't like about this book—as well as what we can do to make our books stronger.

Please note that I cannot help you with technical problems related to the topic of this book, and that due to the high volume of mail I receive, I might not be able to reply to every message.

When you write, please be sure to include this book's title and author as well as your name and phone number or email address. I will carefully review your comments and share them with the author and editors who worked on the book.

Email: feedback@sampublishing.com

Mail: Sams Publishing

800 East 96th Street

Indianapolis, IN 46240 USA

Reader Services

Visit our website and register this book at informit.com/register for convenient access to any updates, downloads, or errata that might be available for this book.

This page intentionally left blank

Introduction

Welcome to *Xcode 4 Unleashed*! This book shows you how to use Apple’s integrated development environment to make great products with the least effort.

Xcode 4 is the descendant of a family of development tools dating back nearly 20 years to NeXT’s ProjectBuilder. It started as a text editor, a user-interface designer, and a front end for UNIX development tools. It has become a sophisticated system for building applications and system software, with a multitude of features that leverage a comprehensive indexing system and subtle incremental parser to help you assemble the right code for your project—and get it right the first time.

That much power can be intimidating. My aim in *Xcode 4 Unleashed* is to demystify Xcode, giving you a gradual tour through examples that show you how you can use it day to day. If you come to Xcode 4 from previous versions, the changes may be overwhelming; Chapter 24, “Xcode 4 for Xcode 3 Veterans,” is just for you, an introduction to where you can find the facilities you’re accustomed to—even newcomers wanting a quick overview may find it useful.

How This Book Is Organized

First, a word on my overall plan for *Xcode 4 Unleashed*. This is a book about developer tools. If it teaches you something about how to use the Cocoa frameworks, or something about programming, that’s fine, but that’s incidental to showing you what Xcode can do.

Every tour needs a pathway, and every lesson needs a story. The first three parts of this book demonstrate Xcode through three applications—a command-line tool, an iOS

IN THIS CHAPTER

- ▶ Introducing Xcode
- ▶ Understanding the goals of *Xcode 4 Unleashed*
- ▶ Considering organization
- ▶ Typographic conventions

app, and a Mac OS X application—that calculate and display some statistics in American football. None of the apps are useful; the graphical apps run almost entirely on sample data, but they demand enough of the development tools to give you a solid insight into how to use them.

The full code for the example programs is available online from www.informit.com/title/9780672333279. In the interest of space, I'll be showing only excerpts.

I'm using applications for iOS and a Mac OS X as examples, but read both Part II, "The Life Cycle of an iOS Application," and Part III, "Xcode for Mac OS X," even if you're interested only in one platform. The applications are only *stories*; the techniques apply to both platforms.

First Steps

In Part I, I'll take you from installing Xcode and running your first project, through basic debugging skills. You'll work through a small command-line application. The application may be simple, but you'll learn foundational skills you'll need before adding the complexity of graphical apps.

- ▶ **Chapter 1, "Getting Xcode"**: Some things to consider before you download Xcode 4; two ways to download and install it.
- ▶ **Chapter 2, "Kicking the Tires"**: Your first look at Xcode, setting a trivial project up and running it.
- ▶ **Chapter 3, "Simple Workflow and Passive Debugging"**: You'll write, build, and run a simple application and respond to a crash.
- ▶ **Chapter 4, "Active Debugging"**: You'll take charge of debugging by setting break-points and tracing through the program. I'll show you how to organize your workspace.
- ▶ **Chapter 5, "Compilation"**: A pause for a description of the process of building an application.
- ▶ **Chapter 6, "Adding a Library Target"**: Add a library target to a project and learn how to build a product from multiple targets.
- ▶ **Chapter 7, "Version Control"**: Why source control is important and how to take advantage of Xcode's built-in support for versioning through Git and Subversion.

The Life Cycle of an iOS Application

Part II tells the story of a small iPhone app and how to use Apple's developer tools to build it. It introduces you to the graphical editor for user interfaces and shows how to profile an app to optimize its speed and memory burden.

- ▶ **Chapter 8, “Starting an iOS Application”:** You’ll start by creating an iOS project and learn the Model/View/Controller design at the core of Cocoa on iOS and Mac OS X alike.
- ▶ **Chapter 9, “An iOS Application: Model”:** Design a Core Data schema and supplement it with your own code.
- ▶ **Chapter 10, “An iOS Controller”:** Create a controller to link your model to the on-screen views. On the way, I’ll tell you about refactoring and Xcode’s continual error-checking.
- ▶ **Chapter 11, “Building a New View”:** You’ll design the user-interface views for your app with the integrated Interface Builder and take advantage of source-code completion.
- ▶ **Chapter 12, “Adding Table Cells”:** While adding an onscreen component to your app, you debug memory management and control how Xcode builds, runs, and tests your apps through the Scheme editor.
- ▶ **Chapter 13, “Unit Testing”:** Unit testing can speed development and make your apps more reliable. I’ll show you how Xcode supports it as a first-class part of the development process.
- ▶ **Chapter 14, “Measurement and Analysis”:** Use Instruments to track down performance and memory bugs.
- ▶ **Chapter 15, “Storyboard”:** Simplifying iOS development with Storyboard, which enables you to specify the whole structure of your app in just one file.
- ▶ **Chapter 16, “Provisioning”:** The three things you must understand to put your app on a device, from testing to the App Store.

Xcode for Mac OS X

Part III shifts focus to Mac OS X development. Some concepts are more important to Mac OS X than iOS, but you’ll be learning techniques you can use whatever your platform.

- ▶ **Chapter 17, “Starting a Mac OS X Application”:** Carrying iOS components over to Mac OS X, Automatic Reference Counting (ARC), and menus and how to link them to the responder chain.
- ▶ **Chapter 18, “Wiring a Mac Application with Bindings”:** As you build a popover window, you’ll use Mac OS X bindings to simplify the link between your data and the screen. With autosizing, you can set up views to resize themselves.
- ▶ **Chapter 19, “A Custom View for Mac OS X”:** Add a custom view to your app to see how Interface Builder can lay it out and configure it, even though it’s not a standard Cocoa component.

- ▶ **Chapter 20, “Localization and Autolayout”**: How you can translate your Mac and iOS apps into other languages. I’ll show you how to use Lion’s autolayout feature to give your views the right size and layout no matter how their contents change.
- ▶ **Chapter 21, “Bundles and Packages”**: You’ll master the fundamental structure of most Mac and iOS products and how both platforms use the `Info.plist` file to fit apps into the operating system.
- ▶ **Chapter 22, “Frameworks”**: Package and share a complete subprogram you can incorporate into any Mac OS X application.
- ▶ **Chapter 23, “Property Lists”**: Learn the basic JSON-like file type for storing data in both Mac OS X and iOS.

Xcode Tasks

Part IV moves on to topics that deserve a more concentrated treatment than Parts II and III.

- ▶ **Chapter 24, “Xcode 4 for Xcode 3 Veterans”**: Written from the point of view of developers moving from Xcode 3’s document-based approach to development to Xcode 4’s “browser” model, this chapter can help you find what you need from Apple’s developer tools.
- ▶ **Chapter 25, “Documentation in Xcode”**: How Xcode gives you both immediate help on API and browsable details on the concepts of Cocoa development. Find out how you can add your own documentation to the system.
- ▶ **Chapter 26, “The Xcode Build System”**: I’ll show you the fundamental rules and tools behind how Xcode goes from source files to executable products.
- ▶ **Chapter 27, “Instruments”**: Using Apple’s timeline profiler, you can go beyond basic performance and memory diagnostics to a comprehensive look at how your program uses its resources.
- ▶ **Chapter 28, “Snippets”**: A roundup of tips, traps, and features to help you get the most from the developer tools.

Appendixes

The appendixes contain references to help you get your bearings on Objective-C, master the build system, choose the right starting points for your project, and find help and support.

- ▶ **Appendix A, “Objective-C”**: A summary of the core programming language for development on Apple platforms, progressing from the simple additions it makes to C, to the specialized features that support Apple frameworks.
- ▶ **Appendix B, “Some Build Variables”**: The most important configuration and environment variables from Xcode’s build system.

- . **Appendix C, “Project and Target Templates”**: Where to start your new projects and files.
- . **Appendix D, “Resources”**: A compendium of books, tools, and Internet resources to support your development efforts.

About Versions

This book was written over most of a year, through the Fall of 2011. In that period Apple issued three major revisions of Xcode, as well as major revisions to both iOS and Mac OS X. *Xcode 4 Unleashed* covers Xcode 4.2, the first version with full support for iOS 5 and Mac OS X 10.7 Lion.

To demonstrate legacy techniques, Part II targets iOS 4.3, with an excursion into iOS 5 for Storyboard. Part III covers a Lion application. Where possible, I've noted changes introduced in Xcode 4.3.

Since the first printing of this book, more versions of Xcode have come out, with new features for Mac OS X 10.8 and iOS 6. We're not leaving you high and dry: Xcode 4 Unleashed: A Supplement is now available and it's free. To access this material, please register this book at informit.com/register and enter the ISBN for the print edition: 9780672333279.

Conventions

This book observes a number of typographic and verbal conventions.

- . Human-interface elements, such as menu items and button labels, are shown **like this**.
- . Filenames and programming constructs are shown like `this`. This will sometimes get tricky, as when I refer to the product of the “Hello World” *project* (plain text, because it's just a noun) as the *file* `Hello World`.
- . Text that you type in will be shown **like this**.
- . A new term is called out *like this*.

You'll do some command-line work in the terminal. Some of the content is wider than this page, so input lines break with backslashes (\) at the ends. Long output lines break simply by splitting them and indenting the continuations. When that output includes long file paths, ellipses (...) replace components, leaving the significant parts.

About the sample code in this book: A significant portion of the code I'll have you write is wrong. As an experienced programmer, you will be in pain and want to correct it. Leave it alone; the errors are intentional.

I'll refer to the location of various development resources as the `/Developer` directory. Xcode 4.3 moved that directory into the Xcode app itself. You can find the additional developer applications in the **Xcode** menu.

For many, many years the Macintosh had a one-button mouse (don't laugh—most purchasers didn't know what a mouse was; try pushing the wrong button on an old Mac mouse). Now it has four ways to effect an alternative mouse click: You can use the right button on an actual mouse; you can hold down the Control key and make an ordinary click; you can hold down two fingers while clicking a multitouch trackpad (increasingly common even on desktop Macs); or you can tap at a designated corner of a multitouch trackpad. And there are more variations as you run through the configurations. Unless the distinction actually matters, the text simply calls for a “right-click,” and you can sort it out.

CHAPTER 8

Starting an iOS Application

Now that you have the basic skills down, let's move on to a real project. You'll build an iPhone application that manages a list of quarterbacks and displays their game and career statistics.

Planning the App

Before coding, it's best (though not customary) to know what you're doing. Specifically, what are you going to present to the app's user, what data do you need to keep to make that presentation, and how do you translate between the data and the presentation?

Model-View-Controller

The Model-View-Controller (MVC) design pattern formalizes those questions into an architecture for graphical applications. The Cocoa Touch application framework is designed to implement applications that follow the MVC pattern. If you don't follow it, you will find yourself "fighting the framework": Winning through to a finished application would be difficult, and maintaining it would be miraculous. The Xcode development tools are designed to support Cocoa programming and therefore the MVC pattern.

MVC divides the functionality of an application into three parts, and each class in the application must fall into one of them:

- ▶ *Model objects* embody the data and logic of a particular problem domain. Models tend to be unique to each application. You can create your own subclasses of `NSObject` or `NSObject` to give life to your models.

IN THIS CHAPTER

- ▶ Planning with the Model-View-Controller pattern
- ▶ Creating an iOS project
- ▶ Using Project Search and Replace

- ▶ *View objects* handle user interaction, presenting information and enabling the user to manipulate data or otherwise influence the behavior of the program. Views are usually drawn from a repertoire of standard elements, such as buttons, tables, scrollers, and text fields. Views ideally know nothing about any problem domain: A button can display itself and report taps without needing to know what tapping means to your application. In iOS, views are instances of `UIView` or its many subclasses.
- ▶ *Controller objects* mediate between the pure logic of the model and the pure mechanics of the views. A controller object decides how views display and how user actions translate into model events. In iOS, controllers are almost always instances of subclasses of `UIViewController`.

NOTE

Okay, in practice some classes won't fall exactly into model, view, or controller. If you have a view custom-built to display your particular data, making that view completely independent of your data model is foolhardy. Still, MVC is an important discipline: If you fudge on it, you should be aware that you're fudging and consider whether you can restore the MVC separation.

The Model

From the nature of a passer rating, all you need is one model class: a `Passer` to carry one passer's name, and the total attempts, completions, yards, touchdowns, and interceptions. Let's make this a little more interesting: Ratings can be calculated over as many attempts as you like and are usually calculated per-game as well as in a career aggregate. So `Passer` should "own" any number of `Game` objects, with details of the game (who played, what date, and so on) as well as the passing statistics for that game.

The model then looks like the diagram presented in Figure 8.1.

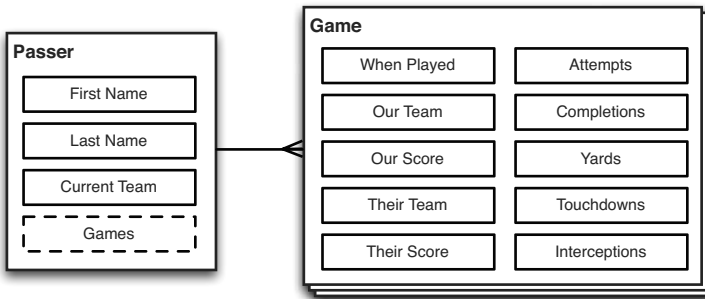


FIGURE 8.1 The summary description of what data a passer-rating app would need leads to the plan shown in this diagram: A `Passer` object serves only to identify a single player; his career statistics are in a set of `Game` objects that `Passer` "owns."

What about the Passer's aggregate statistics—the career yards, touchdowns, and rating? Those can be pulled out of his Games—it turns out not to be hard at all.

The Views

iPhone applications don't usually have a concept of documents, but even simple ones acquire many screens' worth of views. You'll deal in passers and their games, and you need to view and edit both.

So you need a list of passers, who can be created or edited in a separate view; and a view devoted to a selected passer, with a list of games that need a view of their own to create or edit them. A sketch of the flow appears in Figure 8.2.

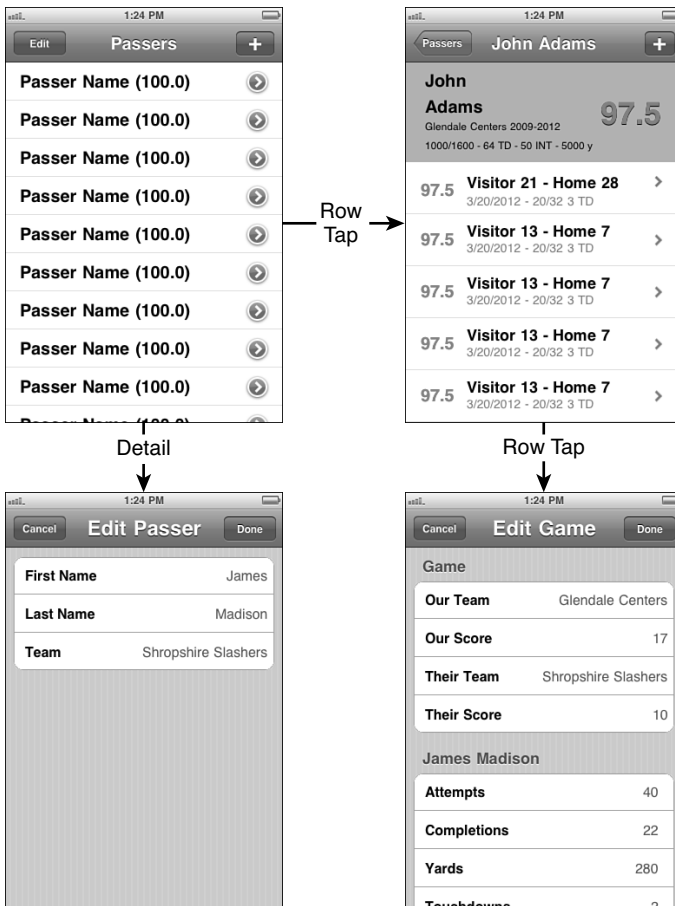


FIGURE 8.2 A rough sketch of the Passer Rating application. The initial view (top left) simply lists all the passers known to the app. Touching the + button creates a passer; touching the detail-disclosure button (>) on a passer opens it for editing. Either way, you use the view at the bottom left to edit it. Touching a passer opens his career record and a list of his games (top right). Games, too, can be added (+) or edited (touch the game), and the app needs an editor for them.

NOTE

That's what a full version of Passer Rating should look like, and getting it down on paper—I used the excellent Blueprint app for the iPad—is an essential step. Alas, this book will run out of Xcode examples before we complete the app.

Typically, each phase of an iPhone application displays a view object—a `UIView`—that fills the screen. That main view usually contains a hierarchy of other views. For instance, the Passer editor at the lower-left corner of the sketch (refer to Figure 8.2) consists of a wrapper `UIView`; it contains a navigation bar (`UINavigationController`, at the top) which in turn contains two buttons (`UIBarButtonItem`, at either end). It also contains a table (`UITableView`) with three rows (`UITableViewCell`), each containing a label (`UILabel`) and a text-entry field (`UITextField`).

The Controllers

iPhone applications are organized around a sequence of *view controllers*, objects derived from `UIViewController`. Each view controller mediates between model objects and the views that fill the iPhone's screen. For each full-screen view you see in the sketch, you must provide a `UIViewController` subclass to link the data in the model to the views on the screen. In the life cycle of a view, the controller comes first; when it is initialized, it creates or loads the view objects and sets them up to reflect the model.

Now, even a simple application like this slides four main views onto and off of the screen, according to a precise hierarchy. Managing the relationships between them—which to slide in, which to slide back to—would seem to be an involved task—and is. But, thankfully, it is not a task you need to worry much about. `UIKit`, the user-facing part of iOS, provides umbrella view controllers (such as `UINavigationController`) that manage the navigation tasks for you by taking ownership of your controller objects. All you need to do is request a transition between the views, and the umbrella takes care of the rest.

NOTE

If you can restrict your project to iOS 5.0 or later, Xcode makes this process even easier. See Chapter 15, “Storyboard.”

Starting a New iPhone Project

Start by creating a new Xcode project, selecting **File**→**New**→**Project...** (⇧⌘N). Select **Application** under **iOS**, and from the array of application types, select **Master-Detail Application**. Passer Rating follows the common pattern of presenting a progression of lists and detail views, under a navigation bar that provides a “breadcrumb” trail back up the tree. The Master-Detail Application template is a skeleton for such an app. Click **Next**. (For more on Xcode's project templates, see Appendix C, “Project and Target Templates.”)

The next panel in the New Project assistant lets you name the project **Passer Rating**. That much is obvious. The next item is **Company Identifier**. Every application in the iOS

universe has a string that uniquely identifies it. The app needs one. The way to ensure uniqueness is to select a name that's unique for you (ordinarily the product name) and then to prefix an inverse-dns string. I own the `wt9t.com` domain, so I'd fill in `com.wt9t`.

NOTE

You don't have a domain name of your own? Next you'll be telling me you don't have a T-shirt for the project. Get one (a domain name). They're cheap, and you don't have to do anything else with them.

Xcode generates an identifier for you and displays it just under the company ID: `com.wt9t.Passer-Rating`.

Next is **Class Prefix**. This is a string that will be prepended to the class names and associated files that the project template creates for you. If this isn't brief, you'll end up with unwieldy names, so try **PR**.

Device Family determines whether the template should include UI setups for iPhone, iPad, or both. Select **iPhone**.

There are four more options. Take two of them:

- ▶ **Use Core Data:** Core Data is Cocoa's object-persistence and relational framework, which is handy for keeping the database organized. The project template will add a number of convenient housekeeping methods for getting a Core Data-based application running.
- ▶ **Include Unit Tests:** Unit testing, the practice of automatically exercising your code to assure yourself that it works, is an important technique in modern software development. Chapter 13, "Unit Testing," covers unit testing in detail, but if Xcode's going to set it up from the start, you're going to take it up.

The other two, **Use Storyboard** and **Use Automatic Reference Counting**, take advantage of features introduced in iOS 5. Storyboard enables you to design not only the user interface for your app, but also how it transitions from view to view as the user navigates through it (see Chapter 15). Automatic Reference Counting (ARC) takes care of memory management in a transparent and reliable way. (It's available in iOS 4.3 but not fully functional.) Both are great features, but you need to target iOS 4.3.

When I get to a Mac version of Passer Rating in Part III, "Xcode for Mac OS X," I'll talk about ARC. For details on reference-counted memory management, see Appendix A, "Objective-C."

Click **Next**. You get a get-file sheet to select a directory to receive the project. Pick something convenient, such as your Desktop directory; that's where Xcode will put the new Passer Rating project directory. And because Chapter 7, "Version Control," sold you on it, check **Create local git repository for this project**. Click **Create** and look at the project Xcode has set up for you.

Target Editor

The Passer Rating project consists of two targets: “Passer Rating,” which produces the app, and “Passer RatingTests,” which will contain the application test suite. Xcode now shows you the Target editor. (Click the Passer Rating project item at the top of the Project navigator to bring it up yourself.) It provides an interface for the basic settings that identify your project—its identifier, target environment, orientations, and the front-end images (icons, startup screen) that are the face UIKit puts on your app.

The **Summary** tab comes in three sections:

- ▶ **iOS Application Target** shows the application identifier; Xcode lets you edit the prefix, but you’re stuck with `Passer-Rating` until you change the product name. You can set a version number (the release version, without markers for alpha or beta status; those confuse Xcode’s packaging process) and a build number (which is where you’d mark prerelease status). And you designate a target (iPhone, iPad, or both) and the “deployment target” (the lowest version of iOS the app can run on). Xcode obligingly sets this to the latest version Apple sells, but for Passer Rating, set it to **4.3**.

Selecting the software development kit (SDK) is as important as the deployment target, even though the setting is in the **Build Settings** tab and not **Info**. This selects the libraries and headers for the highest version of the OS for which you are developing. If the SDK is set for a later version than the deployment target (it must never be set earlier), the features of the later OS are available, but linked “weakly,” so your code can determine at run time whether the features are actually available before trying to use them.

- ▶ **iPhone/iPod Deployment Info** enables you to set the app icons and the launch images (the static PNG the OS shows the user while the app starts up). Also, you have your choice of the orientations you support, presented as a toggle button for the ones you can handle. This sets the *most* orientations the app will tolerate; your individual views may be more restrictive. You can ignore the **Main Interface** combo box; it tells the OS the NIB file (I’ll get to them soon) that specifies your initial UI. For the master-detail app template, you’re set up with code that creates the initial UI, and there’s no need for a NIB to specify the details. Leave this (and **Main Storyboard**) blank.

If you select **iPad** with the **Devices** pop-up, you get an “iPad Deployment Info” section, laid out for iPad resources, instead. And if you select **Universal**, you get both sections.

- ▶ **Linked Frameworks and Libraries** is an editable list of objects that Passer Rating links to: UIKit, Foundation, and Core Data. No need to change it.
- ▶ **Entitlements** sets up options for compatibility with iCloud and shared access to the cryptographic keychain. You probably don’t need those things, and by the time you do, you’ll know what these settings mean. I won’t be covering entitlements.

The **Summary** tab is just a front end for the more comprehensive settings under the **Info** tab, which in turn is a specialized editor for the `Info.plist` file. Chapter 21, “Bundles and Packages,” covers the contents of `Info.plist` exhaustively, but it will be quite some time before you need to change anything.

Copyright, Again

If you browse through the source files, you can see that Xcode has once again used `__MyCompanyName__` as the copyright holder (unless you set a company name in your card in the Address Book application). This has already been covered in the “`__MyCompanyName__`” sidebar in Chapter 7. Again, you have two tasks:

- ▶ Make sure Xcode doesn’t repeat the mistake in this project. Do this by exposing the Utility area (click the right segment of the **View** control) and its File (first tab) inspector, selecting the project (top line) in the Project navigator, and filling in the **Organization** field.
- ▶ Correct the existing files with a projectwide search and replace. This will be your first look at the Search navigator.
 1. Click the third tab (a magnifying glass) in the Navigator area. You see a text field labeled **Find**.
 2. Type `__MyCompanyName__` in the field; a menu pops up offering you options for the search, but the defaults (in-project, containing, match case) will do fine. See Figure 8.3, top.
 3. Press Return. The list under the search field fills with references to every file in which `__MyCompanyName__` appears, with subentries showing the found text in context. If you click one, the editor area displays the file, highlighting the match. Refer to Figure 8.3, middle.
 4. The Search navigator’s **Find** label is a pop-up menu; switch it to **Replace** to expose the replacement field.
 5. The magnifying glass icon in the search field is a drop-down menu containing your search history. The first item, **Show Find Options**, lets you customize the search: literal or regular-expression; restrictions on partial-or whole-word searches; whether it’s case-sensitive; and whether it extends to included frameworks.
 6. Type **Fritz Anderson** in the replacement field and click **Replace All**. You have to do the projectwide search first, or the replace buttons won’t be enabled—Xcode forces you to look at what you’re changing.

Figure 8.3 shows the whole process.

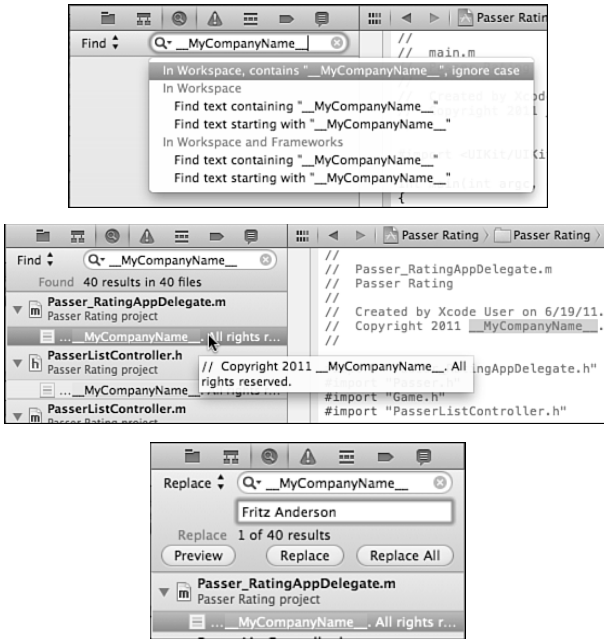


FIGURE 8.3 Expose the Search navigator to start a projectwide search and replace. (Top) Type the search string in the field; Xcode offers some common search options, but simply typing the string and pressing Return usually works. (Middle) When you press Return the navigator fills with every file that contains the string, showing each match in context. Selecting a match puts it into the Editor area. (Bottom) Use the Find/Replace pop-up to expose the replacement field. Type in your replacement string and click **Replace All** to make the change in all files.

At this point, Xcode offers to take a *snapshot* of your project. Snapshots are a supplement to version control; they copy the state of the project into a disk image before you make wholesale changes. Snapshots aren't a replacement for version control or backups—think of them as providing a mass “undo” that doesn't break your workflow as much as a commit would. Look for your snapshot history in the **Projects** panel of the Organizer (**Window**→**Organizer**, ⌘⇧2). From there, you can “Export” the contents of a snapshot to a new directory.

The first snapshot takes a few moments, but after that, they're cheap, so click **Enable**.

At last, you are rewarded with all instances of `__MyCompanyName__` replaced with “Fritz Anderson.” If you switch back to the Project navigator, you see that all the changed files' icons are dark, meaning they have unsaved changes. If you save them all (**File**→**Save All**, ⌘⇧S—you must hold the Option key down to see it in the menu), the “M” version-control badge will appear next to all of them. Now would be a good time to commit a revision to the repository.

The project template you chose for Passer Rating includes a lot:

- ▶ **Frameworks:** The Frameworks group contains UIKit, Foundation, CoreData, and SenTestingKit (which is used only by the unit-test target). They provide links into the iOS system software, and Passer Rating won't run without them. If you open the **Build Phases** tab of the Project editor, for the Passer Rating target, you'll find the frameworks in the Link Binary with Libraries phase.
- ▶ **Class PRAppDelegate:** The application is represented by an object of class UIApplication, which you should never need to subclass or replace. True to the delegation pattern used throughout Cocoa, all the unique behavior of the application comes through the methods of a delegate object, which the application object calls into. PRAppDelegate is declared as a subclass of NSObject and an implementor of the UIApplicationDelegate protocol. The template for the implementation (.m) file contains a good starter for managing the application life cycle, including setting up the Core Data database.
- ▶ **Class PRMasterViewController:** This is, as the name says, the controller for the bottom-level view of Passer Rating, which the design says is a table of passer names and ratings. Because navigation-based applications almost always start with a table, the template makes PRMasterViewController a subclass of UITableViewController, which is suited for running a table view. The implementation file includes skeletons of the methods you need to fill the table in. It also provides an instance of NSFetchedResultsController, which does a lot to help link tables to Core Data data stores.
- ▶ **Class PRDetailViewController:** The controller for the next layer of Passer Rating, the one that is seen when the user taps a passer's name. The template can't be sure what you'll be doing with PRDetailViewController, so it declares it to be an instance of the more-generic UIViewController.
- ▶ **Passer_Rating.xcdatamodeld:** Core Data isn't a full-service relational database (although it uses the SQLite database library internally), but if it were, the Data Model file would be the equivalent of an SQL schema. It defines the entities (think "tables") that hold the data, and the attributes (think "columns") those entities have. Xcode provides a graphical editor for data models.
- ▶ In the Supporting Files group, **Passer_Rating-Info.plist:** This is the source file that yields an Info.plist file to be embedded in the application. It provides basic information on what the application can do, what data it can handle, and how it is presented to the user in the Home screen. Some of that information is presented to the user as text, so its content is merged with the application's InfoPlist.strings for the user's language. (Chapter 20, "Localization and Autolayout," covers localization in Mac OS X, but most of the concepts apply to iOS, as well.)
- ▶ **Miscellaneous source files:** main.m is the standard container for the main() function where the program starts; you usually won't change it. Passer_Rating-Prefix.pch contains common initialization for the compiler.

- ▶ **XIB files:** XIB files are editable files that will be compiled into NIB files and installed in the application. They are archives, mainly of user-interface objects. `PRMasterViewController.xib` contains the `UITableView` for the root view and links it to the `PRMasterViewController` that runs the table. `PRDetailViewController.xib` is a more generic layout that does the same for `PRDetailViewController`.
- ▶ The “Passer RatingTests” target has a class file and `Info.plist` support similar to the app target’s.

Xcode’s template for the project also includes a panoply of build settings, specifying how Passer Rating is to be compiled, linked, and organized.

The project is fully functional, as far as it goes—run it: **Product**→**Run**(⌘R). Xcode builds the app, and in a few seconds, the iOS Simulator starts and launches Passer Rating. Out-of-the-box, the app is the iOS/Core Data equivalent to “Hello World.” It shows an empty table under a navigation bar with **Edit** and **+** buttons. Tapping the **+** button adds a row with the current date and time; tapping the new entry pushes the “detail” view into view; the **Edit** button in the root list (or swiping across a row) lets you delete rows. You can close and reopen the app to find that the rows you added are still there. See Figure 8.4.



FIGURE 8.4 The skeletal code that comes with the Core Data + Master-Detail Application project template is enough to produce an app that can run in the iOS Simulator. It can add rows to its table, and, as shown here, respond to the **Edit** button by offering to delete rows.

NOTE

As to whether Passer Rating is actually saving those dates and times, make sure there's nothing up your sleeve. Closing an iOS app usually doesn't stop it: The OS just puts it to sleep in the background, and its data won't be disturbed. To see a real, fresh restart, go to Xcode, click the **Stop** button in the project toolbar, and then click **Run**.

One More Thing

Passer Rating is supposed to calculate passer ratings, which is a problem you already solved. Let's add your solution to the project. Select **File** → **Add Files to "Passer Rating"...** (⌘⇧A). Xcode opens a modified get-file sheet (Figure 8.5). Track down `rating.h` and `rating.c` from the old passer-rating project and command-click them to select both. Don't click **Add** yet because there's more to do.

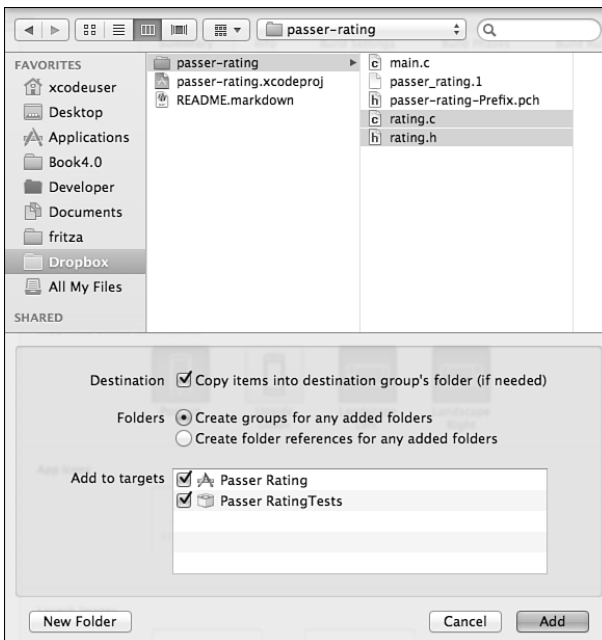


FIGURE 8.5 When you add files to a project, you have some options controlling whether to use the files in-place and how to use them.

Check the box labeled **Copy items into destination group's folder (if needed)**. That adds copies of the files to Passer Rating's project directory. It sometimes makes sense not to do this, as when you want to share a common copy of files across projects, but for now, this is the least complicated way to go. You'll see a better way to do the same thing later.

Under **Folders**, you can decide how Xcode treats any directories you add. It can represent them in the Project navigator by group folders, as an organizational aid, or it can have the project refer to the directories themselves as the objects the project contains. The latter makes sense if you want to include a whole directory of data files in your product. You're not adding any directories, so this isn't an issue, but it's a good idea to make sure the setting is **Create groups for any added folders** because it's usually the safer action to take.

Now you can click **Add**.

Summary

In this chapter, you began work on an iOS application. Before doing anything in Xcode, we decided what the app would do and what it would look like.

When that was done, you knew enough to have Xcode create the project. You explored the Project editor and saw how it managed the configuration of the application. Then you explored the files Xcode's template provided and used the Search navigator to track down and simultaneously repair a bad copyright notice in seven files.

The "empty" application Xcode provided was runnable; you found that running it launched the iOS Simulator, which allowed you to use the app.

Finally, you prepared the app for your own work by adding the `passer_rating` function from the `passer-rating` project.

Now you're ready to make the empty app your own to start implementing the design you put together in this chapter. We'll start with the model.

Index

Symbols

- / (slash), 67
- @author keyword, 380
- @b keyword, 380
- @bug keyword, 380
- @c keyword, 380
- @deprecated keyword, 380
- @dynamic directive in Objective-C, 460
- @e keyword, 380
- @em keyword, 380
- @end in Objective-C, 458
- @exception keyword, 380
- @i keyword, 380
- @implementation directive in Objective-C, 460
- @p keyword, 380
- @param keyword, 380
- @property directives in Objective-C, 458
- @return keyword, 380
- @see keyword, 380
- @synthesize directive in Objective-C, 460
- @todo keyword, 380
- @warning keyword, 380

A

- Accessorizer, 505
- accessors, memory management, 463-464
- ACTION build variable, 477
- active debugging, 35
 - Behaviors panel, 40-42
 - breakpoints, setting, 36-37
 - stepping through code, 38-40
 - variables pane, 37-38
- Activity Monitor instrument, 427
- Activity Monitor instrument template, 433
- Add Files to command (File menu), 60
- Add Item command (Editor menu), 342-343
- Add Remote button, 72
- Add Target command, 57
- Added file state, 70
- adding table cells, 143-146
- Address Book Action Plug-in template, 491
- adjusting build configurations, 396-397
- ADP (Automatic Device Provisioning), 223-225
- allocations (memory), tracking, 182-187
- Allocations instrument, 182-187, 426, 444
- Allocations instrument template, 433

- AllowFileAccessOutsideOfWidget key (Info.plist), 323
- AllowInternetPlugins key (Info.plist), 323
- AllowJava key (Info.plist), 323
- AllowNetworkAccess key (Info.plist), 324
- AllowSystem key (Info.plist), 324
- analysis
 - analyzer, 193-195
 - Automatic Reference Counting (ARC), 195-196
 - memory, 182
 - allocations, 182-187
 - leaks, 187-189
 - zombies, 189-193
 - speed, 173-182
- Analyze command (Product menu), 54, 193-195
- analyzer, 193-195
- APFiles key (Info.plist), 319
- app IDs, 227-228
- App Store releases, preparing, 234. *See also* provisioning
 - iTunes Connect, 234-235
 - validating and submitting, 235-236
- AppKiDo, 505
- AppKiDo-for-iPhone, 505
- Apple Developer Forums, 500
- Apple developer programs, 12-13, 502
- Apple Developer Technical Support (DTS), 501-502
- application bundles, 309-311
- application IDs, 223, 227-228
- application plug-in templates, 491
- Application template, 495, 497
- application templates
 - iOS project templates, 487-488
 - Mac OS X project templates, 489-490
- application testing, 156, 166-168
- applications (Mac OS X)
 - application bundles, 309-311
 - ARC (Automatic Reference Counting), 246-248
 - bindings, 253
 - Game popover example, 269-273
 - Game table example, 268-269
 - Passer table example, 266-268
 - running, 260-263
 - Team table example, 260
 - building, 332
 - creating, 240-243
 - custom views, 275-276

- NSView subclass, creating, 276-279
 - properties, 282-283
 - view controller, 279-282
 - files, 241-242
 - frameworks
 - adding to builds, 327
 - debugging, 332-335
 - explained, 325
 - framework targets, 326
 - installing, 327-329, 331
 - location of, 330-331
 - populating, 326-327
 - goals, 239-240
 - Info.plist. *See* Info.plist file
 - LeagueDocument.m, loading data into, 250-251
 - localization, 285-286
 - adding, 286
 - Info.plist, 300-302
 - MainMenu.xib, 288-291
 - strings in code, 302-306
 - testing, 287-288
 - window XIBs, 291-300
 - menus, adding, 248-250
 - model, porting from iOS, 243-246
 - object controllers, 258-260
 - popover windows, 269-273
 - project options, 240-241
 - property lists
 - binary property lists, 348-349
 - creating, 341-345
 - editing, 338-341
 - explained, 337
 - Property List editor, 345-347
 - specialized property lists, 349-350
 - text property lists, 348
 - types of, 337-338
 - views
 - Autoresizing, 255-257
 - laying out, 263-264
 - table view, 254-255
- apps (iOS), 221. *See also* projects**
- App Store releases, preparing. *See* provisioning
 - Automatic Reference Counting (ARC), 195-196
 - Info.plist. *See* Info.plist file
 - memory, 182
 - allocations, 182-187
 - leaks, 187-189
 - zombies, 189-193
 - planning, Model-View-Controller (MVC) design pattern, 87-90
 - provisioning. *See* provisioning
 - speed, measuring, 173-182
- ARC (Automatic Reference Counting), 146, 195-196, 466-467**
- Mac OS X applications, 246-248
 - in Xcode 4, 367
- archive action (xcodebuild), 401**
- ARCHS build variable, 480**
- ARCHS_STANDARD_32_64_BIT build variable, 480**
- ARCHS_STANDARD_64_BIT build variable, 480**
- Ash, Mike, 502**
- assemblers, 55**
- assembly code, 46, 48**
- Assembly File template, 496**
- assertions, SenTestingKit, 168**
- equality tests, 169
 - exceptions, 169-170
 - simple tests, 169
- assistant editor, 355-356, 441-442**
- Assistant Editor command (View menu), 356**
- ATSApplicationFontsPath key (Info.plist), 316**
- Attach to Process command (Product menu), 332**
- attribute accessors, memory management, 463-464**
- attributes**
- in KVC (Key-Value Coding), 461
 - for Passer Rating project, 100-102
- autolayout, 292-296**
- horizontal spacing, 297-300
 - vertical spacing, 297
- automatic code completion, 139-141**
- Automatic Device Provisioning (ADP), 223-225**
- Automatic Reference Counting (ARC), 146, 195-196, 466-467**
- Mac OS X applications, 246-248
 - in Xcode 4, 367
- Automation instrument, 429**
- Automation instrument template, 434**
- Automator Action template, 491**
- Autoresizing, 255-257**

B

- Bare Bones Software, 504**
- BBEdit, 504**
- behavior keys (Info.plist), 322**
- behaviors, 362-363**
- Behaviors panel, 40-42**
- Big Nerd Ranch, 503**
- binary files, creating universal binaries, 410**

- binary property lists, 348-349
 - bindings, 253
 - Game popover, 269-273
 - Game table, 268-269
 - Passer table, 266-268
 - running, 260-263
 - Team table, 260
 - Blame view, 81-82
 - Blank instrument template, 433
 - blogs, 502-503
 - Bluetooth instrument, 429
 - bookmarks, documentation, 375
 - books, 499-500
 - branching, 82-84
 - breakpoints, 36-37, 443
 - Breakpoints button, 36
 - Breakpoints editor, equivalent in Xcode 4, 363
 - Breakpoints navigator (Xcode 4), 363
 - browser (Xcode 4), 353-355, 364
 - browsing documentation, 373-374
 - Buck, Erik, 499
 - build action (xcodebuild), 400
 - Build command (Product menu), 30, 118
 - build configurations, 396
 - adding, 398
 - adjusting, 396-397
 - build errors, 28-30
 - build logs, 403-404
 - Build New Instrument command (Instrument menu), 432
 - build phases, 58, 390-391
 - Build Phases tab (Target editor), 49
 - build process
 - build configurations, 396-398
 - build logs, 403-404
 - build phases, 390-391
 - Build Settings, 393-395, 438
 - build transcript example. See build transcript example
 - build variables. See build variables
 - configuration files (.xcconfig), 398-400
 - custom build rules, 401-403
 - explained, 389-392
 - makefiles, 389-390
 - targets, 390
 - in Xcode 4, 358
 - xcodebuild tool, 400-401
 - Build Results window, equivalent in Xcode 4, 363
 - Build Settings, 393-395, 438
 - build targets, 477-478
 - build transcript example, 404-405
 - compiling source files, 408-409
 - creating universal binaries, 410
 - linking, 409
 - precompiled header, 407
 - resources, 406
 - touch tool, 410
 - build variables, 473-475
 - build targets, 477-478
 - bundle locations, 479-480
 - compiler settings, 480-481
 - deployment, 482
 - destination locations, 478-479
 - editing, 395-396
 - environment, 475-477
 - explained, 392-393
 - Info.plist, 482-483
 - settings, 474-475
 - settings hierarchy, 393-395
 - source locations, 478
 - source trees, 483
 - building
 - Mac Passer Rating, 332
 - projects, 21-23
 - build errors, 28-30
 - passer-rating project, 28-30
 - builds
 - adding Doxygen to, 386-387
 - Build Settings, 438
 - tips and tricks, 445-447
 - BUILT_PRODUCTS_DIR build variable, 479
 - Bumgarner, Bill, 444, 503
 - bundle location build variables, 479-480
 - Bundle template, 490
 - bundles. See also packages
 - application bundles, 309-311
 - definition of, 309
 - examples of, 309
 - Info.plist file, 311-312
 - keys, 312-324
 - localization, 300-302, 312
- ## C
- C programming language
 - iOS file templates, 494-495
 - Mac OS X file templates, 496
 - relationship with Objective-C, 456
 - C++ programming language
 - iOS file templates, 494-495
 - Mac OS X file templates, 496
 - Objective-C++ and, 471
 - C++ Standard Template Library, 469
 - C/C++ Library template, 491

Callahan, Kevin, 505

Carbon Events instrument, 430

cells, table

adding, 143-146

creating custom, 149-153

certificates

Development Certificates, 225

Distribution Certificates, 225-226

CFAppleHelpAnchor key (Info.plist), 316

CFBundleDevelopmentRegion key (Info.plist), 314

CFBundleDisplayName key (Info.plist), 314

CFBundleDocumentTypes key (Info.plist), 314

CFBundleExecutable key (Info.plist), 312

CFBundleGetInfoString key (Info.plist), 313

CFBundleHelpBookFolder key (Info.plist), 316

CFBundleHelpBookName key (Info.plist), 316

CFBundleIconFile key (Info.plist), 313

CFBundleIconFiles key (Info.plist), 320

CFBundleIdentifier key (Info.plist), 313

CFBundleInfoDictionaryVersion key (Info.plist), 313

CFBundleLocalizations key (Info.plist), 315

CFBundleName key (Info.plist), 314

CFBundlePackageType key (Info.plist), 313

CFBundleShortVersionString key (Info.plist), 313

CFBundleSignature key (Info.plist), 313

CFBundleURLTypes key (Info.plist), 315

CFBundleVersion key (Info.plist), 313

CFPlugInDynamicRegisterFunction key (Info.plist), 322

CFPlugInDynamicRegistration key (Info.plist), 322

CFPlugInFactories key (Info.plist), 322

CFPlugInTypes key (Info.plist), 322

CFPlugInUnloadFunction key (Info.plist), 322

checking connections, 137

Citation class in Objective-C, 458

Clang front-end parser, 466

Class Browser, absence in Xcode 4, 364

class extensions, 145

class implementation in Objective-C, 458-460

class interfaces in Objective-C, 457-458

class methods in Objective-C, 458

Class Model window, absence in Xcode 4, 364

class names, refactoring, 114-115

classes, 503

GameTableCell, 207-210

managed-object classes

creating, 105-106

extending, 106-108

names, refactoring, 114-115

NSBundle, 285

NSFetchedResultsController, 116

PasserListController, 114

PRAppDelegate, 95

PRDetailViewController, 95

PRMasterViewController, 95, 114

PRSPasserEditController, 212-215

UILabel, 132

UIStoryboardSegue, 200

UIViewController, 90

clean action (xcodebuild), 400

Clean command (Product menu), 23

cloning, 74-75

CloseBoxInsetX key (Info.plist), 323

CloseBoxInsetY key (Info.plist), 323

closing Workspace window, 23

Cocoa

Mac OS X file templates, 496

Objective-C 2.0 and, 460

dynamic dispatch, 470-471

fast enumeration, 467-468

Foundation data types, 468-469

Key-Value Coding (KVC), 461-462

memory management, 462-464

properties, 464-467

version support, 461

Cocoa-AppleScript Application template, 489

Cocoa Application template, 489

Cocoa Design Patterns, 499

Cocoa Design Patterns (Buck), 499

cocoa-dev mailing list, 501

Cocoa Events instrument, 430

Cocoa Framework template, 490

Cocoa-in-Java, 455

Cocoa Layout instrument, 425

Cocoa Layout instrument template, 435

Cocoa Library template, 490

Cocoa Literature, 502

Cocoa Programming for Mac OS X (Hillegass), 499

Cocoa Samurai, 503

Cocoa Touch, iOS file templates, 493-494

Cocoa Touch Static Library template, 488

xCocoaBuilder, 501

CocoaDev wiki, 502

CocoaHeads, 503

code completion, 139-141

Code completion options (Preferences), 21

Code Folding Ribbon, 440-441

code snippets, 144

Command Line Tool template, 490

command-line tools. *See names of specific tools*

commands. *See names of specific commands*

Commit command (Source Control menu), 70

commits, 70-71

Comparison view, 79-81

compiler products

- intermediate products, 55
- object files, 50, 55
- precompiled headers, 56

compiler settings build variables, 480-481**compilers**

- compiler products, 50, 55-56
- definition of, 46
- gcc, 52, 367
- llvm, 52-54
- llvm-gcc, 54

compiling

- compiler products, 50, 55-56
- dynamic loading, 51-52
- explained, 45-49
- gcc compiler, 52
- linking, 50-51
- llvm compiler, 52-54
- llvm-gcc compiler, 54
- predictive compilation, 54
- source files, 408-409

CONFIGURATION build variable, 477**configuration files (.xconfig), 398**

- creating, 398
- preprocessing, 399-400
- SDK- and architecture-specific settings, 399

Configuration Settings File template, 496**configurations (builds), 396**

- adding, 398
- adjusting, 396-397

configureCell:atIndexPath: method, 116-117**configuring**

- Doxygen
 - with Expert tab, 383-384
 - with wizard, 381-383
- Instruments, 420-421

Conflicted file state, 68, 70**conflicts, 75-79****connecting**

- GameListController, 137-139
- tables to view controllers, 143-146

Connection inspector, 250**connections, checking, 137****Connections inspector, 137****CONTENTS_FOLDER_PATH build variable, 479****controllers, 90, 113**

- editing, 116
 - Passers, creating, 117-118
 - table view, 116-117
- editor controllers, 212-215
- GameListController, connecting, 137-139
- object controllers, Team Array Controller, 258-260

PasserGraphController, 279-282

- table-view controllers, 143
- view controllers, connecting to tables, 143-146

Core Animation instrument, 426**Core Animation instrument template, 434****Core Data**

- iOS file templates, 495
- Mac OS X file templates, 497

Core Data Cache Misses instrument, 424**Core Data Faults instrument, 425****Core Data Fetches instrument, 425****Core Data instrument template, 435****Core Data Saves instrument, 425****Core Foundation, 469****Counters instrument, 428****Counters instrument template, 435****CPU Activity instrument, 429****CPU Monitor instrument, 427****CRLF pair line endings in CSV files, 163****CSResourcesFileMapped key (Info.plist), 316****CSV reader, testing, 159-166****custom build rules, 401-403****custom instruments, 431-432****custom table cells, creating, 149-153****custom table view cell, 207-210****custom views for Mac OS X, 275-276**

- NSView subclass, creating, 276-279
- properties, 282-283
- view controller, 279-282

CVS, 367**D****dashboard widgets, Info.plist keys, 323-324****Data Model template, 495****data models. See models****data sets for logic tests, 158-159****data types, Foundation data types in****Objective-C 2.0, 468-469****dealloc method, 460, 463****Debug area, 22-23****DEBUG macro, 451****Debug menu commands, 38-39****Debug navigator, 31, 120-122, 364****debugging**

- active debugging, 35
- Behaviors panel, 40-42
- breakpoints, setting, 36-37
- build errors, 28-30
 - with Debug navigator, 120-122
- dependent targets, 63

- frameworks, 332-335
 - live issues with Fix-it, 118-120
 - memory management, 146-149
 - Passer Rating project, 32-33, 111
 - stack traces, 31
 - stepping through code, 38-40
 - tips and tricks, 443-445
 - variables pane, 37-38
 - defaults command-line tool, 437**
 - deleting. See removing**
 - dependencies, implicit, 63**
 - dependent targets, 62-63**
 - deployment build variables, 482**
 - DEPLOYMENT_POSTPROCESSING build variable, 482**
 - derived-data directory, 478**
 - DERIVED_FILE_DIR build variable, 479**
 - design patterns, Model-View-Controller (MVC), 87-90**
 - desktop (Xcode 4), 353-355**
 - destination location build variables, 478-479**
 - Detail area (trace documents), 417-418**
 - Detail view, absence in Xcode 4, 360**
 - DEVELOPER_APPLICATIONS_DIR build variable, 475**
 - DEVELOPER_BIN_DIR build variable, 476**
 - DEVELOPER_DIR build variable, 475**
 - /Developer directory, 11**
 - DEVELOPER_FRAMEWORKS_DIR build variable, 476**
 - DEVELOPER_LIBRARY_DIR build variable, 476**
 - developer programs (Apple), 12-13**
 - developer programs (iOS), 221-222**
 - DEVELOPER_SDK_DIR build variable, 476**
 - DEVELOPER_TOOLS_DIR build variable, 476**
 - Developer Tools' Group, 22**
 - DEVELOPER_USR_DIR build variable, 476**
 - Development Certificates, 225**
 - development traps, 448-451**
 - device IDs, 226-227**
 - Diagrams panel (Doxygen), 383**
 - directories, 11**
 - Directory I/O instrument, 425**
 - Disk Monitor instrument, 428**
 - Dispatch instrument, 425**
 - Dispatch instrument template, 435**
 - Display Brightness instrument, 429**
 - distribution builds, 231-233**
 - Distribution Certificates, 225-226**
 - distribution profiles, 229-230**
 - docsets (documentation sets)**
 - adding to builds, 386-387
 - downloading, 376-377
 - explained, 375-376
 - installing, 385-386
 - documentation. See also docsets**
 - bookmarks, 375
 - browsing, 373-374
 - Doxygen. See Doxygen
 - help. See help
 - searching, 374-375
 - trace documents. See trace documents
 - Documentation directory, 11**
 - Documentation option (Installer package), 15**
 - Documentation organizer**
 - bookmarks, 375
 - browsing documentation, 373-374
 - searching documentation, 374-375
 - documentation sets (docsets)**
 - adding to builds, 386-387
 - downloading, 376-377
 - explained, 375-376
 - installing, 385-386
 - documents, Info.plist keys, 314-315**
 - Dot panel (Doxygen), 384**
 - downloading docsets, 376-377**
 - Doxygen**
 - adding to builds, 386-387
 - configuring with Expert tab, 383-384
 - configuring with wizard, 381-383
 - generating documentation with, 378-381
 - installing, 378
 - installing docsets, 385-386
 - running, 384-385
 - DSTROOT build variable, 479**
 - DTrace Script Export command (File menu), 432**
 - DTrace Script Import command (File menu), 432**
 - DTS (Apple Developer Technical Support), 501-502**
 - .dylib file extension, 52**
 - dynamic dispatch in Objective-C 2.0, 470-471**
 - dynamic libraries, 52**
 - dynamic loading, 51-52**
- ## E
- Edit All in Scope command, 53**
 - Edit menu commands, Refactor, 114**
 - editing**
 - build variables, 395-396
 - property lists, 338-341
 - view controllers, 116
 - fetchResultsController method, 117
 - Passers, creating, 117-118
 - table view, 116-117
 - Editor control, 355**

Editor menu commands, 437

- Add Item, 342-343
- Embed In, 275-276
- Expand All Transcripts, 404
- Show Raw Values & Keys, 350
- Simulate Document, 255

Editor Style control, 100**editors**

- assistant editor, 441-442
- Passer
 - editor controller, 212-215
 - editor view, 210, 212
- Property List editor, 345-347
- Run Script editor, 109
- special-purpose editors (Xcode 4), 362-364
- text editors, 504-505
- Version editor
 - Blame view, 81-82
 - Comparison view, 79-81
 - Log view, 82
- in Xcode 4
 - assistant editor, 355-356
 - Editor control, 355
 - multiple editors, 356-358

emacs, 504**Embed In command (Editor menu), 275-276****embedded files, referencing in test classes, 166****Empty Application template, 488****Empty template, 488, 492, 495-497****Energy Diagnostics instrument template, 434****Energy Usage instrument, 429****Enterprise program (iOS developer program), 222****entities for Passer Rating project, 100****enumeration, fast enumeration in**

- Objective-C 2.0, 467-468

environment build variables, 475-477**environment variables, setting, 147-149****equality test assertions, 169****errors. See also debugging**

- build errors, 28-30
- definition of, 29

Essentials option (Installer package), 14**EXC_BAD_ACCESS, 32****exceptions, SenTestingKit assertions and, 169-170****EXECUTABLE_EXTENSION build variable, 477****EXECUTABLE_FOLDER_PATH build variable, 480****Executable Info window, equivalents in Xcode 4, 361****EXECUTABLE_NAME build variable, 477****EXECUTABLE_PATH build variable, 480****EXECUTABLE_PREFIX build variable, 477****executables in Xcode 4, 359****Expand All Transcripts command (Editor menu), 404****Expert tab (Doxygen), 383-384****Export Items command (Keychain Access), 226****Exports File template, 497****Extended Detail area (trace documents), 418-419****extending classes, 106-108****External Build System template, 492****F****fast enumeration in Objective-C 2.0, 467-468****fetchResultsController method, 117****fields, searching/replacing (Passer Rating project), 93-94****File Activity instrument, 425****File Activity instrument template, 435****File Attributes instrument, 426****file extensions**

- .dylib, 52
- .m, 50
- .pch, 56

File Info window, equivalents in Xcode 4, 361**File Locks instrument, 426****File menu commands**

- Add Files to, 60
- DTrace Script Export, 432
- DTrace Script Import, 432
- New, 18
- New Project, 90, 240
- Open, 18
- Open Quickly, 371
- Record Options, 423
- Save as Template, 423

file states

- in Git, 68-69
- in Subversion, 68
- in Xcode, 69-70

file system instruments, 425-426**File Template library, 497-498****file templates**

- iOS templates, 493
 - C and C++, 494-495
 - Cocoa Touch, 493-494
 - Core Data, 495
 - resource templates, 495-496
- Mac OS X templates
 - C and C++, 496
 - Cocoa, 496
 - Core Data, 497

- resource templates, 497
- user interface, 497

files

- adding to projects (Passer Rating project), 97-98
- adding to targets, 59-61
- configuration files (.xcconfig), 398-400
- file states
 - in Git, 68-69
 - in Subversion, 68
 - in Xcode, 69-70
- filtering in Xcode 4, 360
- Info.plist. See Info.plist file
- InfoPlist.strings, 326
- LeagueDocument.m, loading data into, 250-251
- main.m, 95
- MainMenu.xib file, localization, 288-291
- makefiles, 389-390
- .mom files, 99
- object files, 50, 55
- PassCompletionView.h, 276-277
- PassCompletionView.m, 277-279
- passer-rating project, 26-28
- PasserGraph.h, 326
- PasserGraph-Info.plist, 326
- PasserGraph.m, 326
- PasserGraph-Prefix.pch, 326
- PasserGraphController.h, 279
- PasserGraphController.m, 280
- prefix files, 56
- renaming, 440
- uninstall-developer-folder, 12
- uninstall-devtools, 12
- .xcdatamodel, 99
- in Xcode 4, 358-360
- XIB files, 96, 128-129, 291-300

fillWithData: method, 250-251

filtering files/symbols in Xcode 4, 360

Fix-it, 118-120

Flatten Recursion display option, 418

folder references, 438-439

Font key (Info.plist), 324

formal protocols in Objective-C, 456

forums, 500-501

Foundation data types in Objective-C 2.0, 468-469

framework & library templates

- iOS project templates, 488
- Mac OS X project templates, 490-491

frameworks, 95

- adding to builds, 327
- debugging, 332-335
- definition of, 56

- explained, 325
- framework targets, 326
- installing, 327-329, 331
- location of, 330-331
- populating, 326-327

FRAMEWORKS_FOLDER_PATH build variable, 480

functional testing. See application testing

G

Game table, binding, 268-269

GameListController, connecting, 137-139

GameTableViewCell class, 207-210

garbage collection in Objective-C 2.0, 462

Garbage Collection instrument, 426

GC Monitor instrument template, 435

gcc compiler, 52, 367

GCC_ENABLE_OBJC_GC build variables, 480

GCC_PREPROCESSOR_DEFINITIONS build variable, 480

GCC_TREAT_WARNINGS_AS_ERRORS build variable, 481

GCC_VERSION build variable, 480

Generate Assembly File command (Generate Output menu), 55

generate-games.rb script, 108-111

Generate Output command (Product menu), 55

Generate Output menu commands, 55

Generate Preprocessed File command (Generate Output menu), 55

generating documentation with Doxygen, 378-381

Generic C++ Plug-in template, 492

Generic Kernel Extension template, 492

getters, memory management, 463-464

Git, 26

Git repositories, creating by hand, 66-68

GitHub, 503

GPS instrument, 429

GPX File template, 495

graphics instruments, 426

GROUP build variable, 475

groups

- renaming, 440
- in Xcode 4, 358-360

H

HEADER_SEARCH_PATHS build variable, 481

headers

- precompiled headers, 56
- in targets, 61

heapshot-analysis table, 444

Height key (Info.plist), 323

Hello World project

- building, 21-23
- creating, 19-21
- running, 21-23

help

- help menu, 372
- Open Quickly dialog, 371-372
- Quick Help inspector, 369-370
- Quick Help popover, 370-371
- Xcode how-to's, 373

help keys (Info.plist), 316**Help menu commands, 372**

- Quick Help for Selected Item, 370
- Search Documentation for Selected Text, 371

Hide Missing Symbols display option, 418**Hide System Libraries display option, 418****HOME build variable, 475****horizontal spacing (labels), 297-300****how-to's, 373****HTML panel (Doxygen), 384****I****I/O Activity instrument, 426****IDs**

- application IDs, 223, 227-228
- device IDs, 226-227

Ignored file state, 68**Image Unit Plug-in template, 492****implicit dependencies, 63****#import header file, 457****Indentation options (Preferences), 21****Individual program (iOS developer program), 222****Info windows, equivalents in Xcode 4, 360-362****Info.plist build variables, 482-483****Info.plist file, 311-312**

- keys. See keys (Info.plist file)
- localization, 300-302, 312

INFOPLIST_EXPAND_BUILD_SETTINGS build variable, 483**INFOPLIST_FILE build variable, 482****INFOPLIST_OUTPUT_FORMAT build variable, 483****INFOPLIST_PREPROCESS build variable, 483****InfoPlist.strings file, 326****install action (xcodebuild), 400****INSTALL_DIR build variable, 482****INSTALL_GROUP build variable, 482****INSTALL_MODE_FLAG build variable, 482****INSTALL_OWNER build variable, 482****INSTALL_PATH build variable, 482****INSTALL_ROOT build variable, 482****Installer package, 13-15****Installer Plug-in template, 491****installing**

- docsets, 385-386
- Doxygen, 378
- frameworks, 327-329, 331
- Xcode, 10-11
 - Installer package, 13-15
 - system requirements, 9-10

installsrc action (xcodebuild), 401**instance methods in Objective-C, 458****instance variables in Objective-C, 458****instantiating templates, 485-487, 498****Instrument menu commands, Build New****Instrument, 432****Instruments**

- Activity Monitor, 427
- Allocations, 182-187, 426, 444
- Automation, 429
- Bluetooth, 429
- Carbon Events, 430
- Cocoa Events, 430
- Cocoa Layout, 425
- configuration, 420-421
- Core Animation, 426
- Core Data Cache Misses, 424
- Core Data Faults, 425
- Core Data Fetches, 425
- Core Data Saves, 425
- Counters, 428
- CPU Activity, 429
- CPU Monitor, 427
- custom instruments, 431-432
- Directory I/O, 425
- Disk Monitor, 428
- Dispatch, 425
- Display Brightness, 429
- Energy Usage, 429
- explained, 411-412
- File Activity, 425
- File Attributes, 426
- File Locks, 426
- Garbage Collection, 426
- GPS, 429
- I/O Activity, 426
- instruments available to iOS, 430-431
- JavaThread, 429
- Leaks, 187-189, 427, 444
- Library palette, 419-420
- Memory Monitor, 428
- Network Activity, 429
- Network Activity Monitor, 428
- Object Graph, 427

- OpenGL Driver, 426
- OpenGL ES Analyzer, 426
- OpenGL ES Driver, 426
- Process, 428
- Reads/Writes, 426
- running, 412-413
- Sampler, 428
- Shared Memory, 427
- Sleep/Wake, 429
- speed, measuring, 174-182
- Spin Monitor, 428
- Sudden Termination, 425
- templates, 433-435
- Thread States, 429
- Time Profiler, 428
- trace documents. *See* trace documents
- User Interface, 426
- VM Tracker, 427
- WiFi, 429
- Interface Builder**
 - tips and tricks, 442
 - in Xcode 4, 365-366
- interfaces, class interfaces in Objective-C, 457-458**
- intermediate compiler products, 55**
- Invert Call Tree display option, 418**
- iOKit Driver template, 492**
- iOS developer programs, 221-222**
- iOS file templates, 493**
 - C and C++, 494-495
 - Cocoa Touch, 493-494
 - Core Data, 495
 - resource templates, 495-496
- iOS Passer Rating project. *See* Passer Rating project (iOS)**
- iOS Programming: The Big Nerd Ranch Guide* (Hillegass), 499**
- iOS project templates**
 - application templates, 487-488
 - framework & library templates, 488
- iOS projects. *See* projects (iOS)**
- iOS Provisioning Portal, 225**
 - application IDs, 227-228
 - Development Certificates, 225
 - device IDs, 226-227
 - Distribution Certificates, 225-226
 - distribution profiles, 229-230
 - Team Provisioning Profile, 228
- Issues navigator (Xcode 4), 363**
- iTunes Connect, 234-235**

J–K

- JavaThread instrument, 429**
- jump bar, 440**
- Key Bindings, 439**
- Key-Value Coding (KVC), 461-462**
- Keyboard panel, 39**
- Keychain Access**
 - Export Items, 226
 - Request a Certificate from a Certificate Authority, 225
- keys, 342**
- keys (Info.plist), 312**
 - AllowFileAccessOutsideOfWidget, 323
 - AllowInternetPlugins, 323
 - AllowJava, 323
 - AllowNetworkAccess, 324
 - AllowSystem, 324
 - APFiles, 319
 - ATSApplicationFontsPath, 316
 - CFAppleHelpAnchor, 316
 - CFBundleDevelopmentRegion, 314
 - CFBundleDisplayName, 314
 - CFBundleDocumentTypes, 314
 - CFBundleExecutable, 312
 - CFBundleGetInfoString, 313
 - CFBundleHelpBookFolder, 316
 - CFBundleHelpBookName, 316
 - CFBundleIconFile, 313
 - CFBundleIconFiles, 320
 - CFBundleIdentifier, 313
 - CFBundleInfoDictionaryVersion, 313
 - CFBundleLocalizations, 315
 - CFBundleName, 314
 - CFBundlePackageType, 313
 - CFBundleShortVersionString, 313
 - CFBundleSignature, 313
 - CFBundleURLTypes, 315
 - CFBundleVersion, 313
 - CFPlugInDynamicRegisterFunction, 322
 - CFPlugInDynamicRegistration, 322
 - CFPlugInFactories, 322
 - CFPlugInTypes, 322
 - CFPlugInUnloadFunction, 322
 - CloseBoxInsetX, 323
 - CloseBoxInsetY, 323
 - CSResourcesFileMapped, 316
 - Font, 324
 - Height, 323
 - LSApplicationCategoryType, 316
 - LSArchitecturePriority, 316
 - LSBackgroundOnly, 317

LSEnvironment, 317
 LSExecutableArchitectures, 317
 LSFileQuarantineEnabled, 318
 LSFileQuarantineExcludedPathPatterns, 318
 LSGetAppDiedEvents, 317
 LSHasLocalizedDisplayName, 315
 LSMinimumSystemVersion, 317
 LSMinimumSystemVersionByArchitecture, 317
 LSMultipleInstancesProhibited, 317
 LSPrefersCarbon, 319
 LSRequiresiPhoneOS, 320
 LSRequiresNativeExecution, 317
 LSUIElement, 318
 LSUIPresentationMode, 318
 LSVisibleInClassic, 319
 MainHTML, 324
 NSAppleScriptEnabled, 318
 NSHumanReadableCopyright, 316
 NSJavaNeeded, 320
 NSJavaPath, 320
 NSJavaRoot, 320
 NSMainNibFile, 314
 NSPersistentStoreTypeKey, 319
 NSPrefPanelIconFile, 323
 NSPrefPanelIconLabel, 323
 NSPrincipalClass, 314
 NSServices, 318
 NSSupportsSuddenTermination, 317
 OSAScriptingDefinition, 318
 Plug-in, 324
 QLNeedsToBeRunInMainThread, 319
 QLPreviewHeight, 318
 QLPreviewWidth, 318
 QLSupportsConcurrentRequests, 318
 QLThumbnailMinimumSize, 318
 SMAuthorizedClients, 319
 SMPrivilegedExecutables, 319
 UIAppFonts, 320
 UIApplicationExitsOnSuspend, 322
 UIBackgroundModes, 322
 UIDeviceFamily, 320
 UIFileSharingEnabled, 322
 UIInterfaceOrientation, 321
 UILaunchImageFile, 321
 UIMainStoryboardFile, 314
 UIPrerenderedIcon, 321
 UIRequiredDeviceCapabilities, 320
 UIRequiresPersistentWiFi, 320
 UIStatusBarHidden, 321
 UIStatusBarStyle, 321
 UISupportedExternalAccessory-
 Protocols, 321

UISupportedInterfaceOrientations, 321
 UIUpgradeOtherBundleIdentifier, 319
 UIViewEdgeAntialiasing, 321
 UIViewGroupOpacity, 321
 UTExportedTypeDeclarations, 315
 UTImportedTypeDeclarations, 315
 Width, 323

Kochan, Stephen, 499
KVC (Key-Value Coding), 461-462

L

labels, 132-134, 151
launch behavior keys (Info.plist), 316-318
laying out views, 263-264
LeagueDocument.m, loading data into, 250-251
leaks (memory), 187-189
Leaks instrument, 187-189, 427, 444
Leaks instrument template, 434
Lee, Graham, 500
libraries
 adding, 63
 definition of, 50
 dynamic libraries, 52
 targets
 adding, 57-58
 adding files to, 59-61
 build phases, 58
 definition of, 58
 dependent targets, 62-63
 headers in, 61
Library palette (Instruments), 419-420
LIBRARY_SEARCH_PATHS build variable, 481
line endings in CSV files, 163
linkage editing, 50-51
linking, 50-51, 409
lists
 Passer list, reconstructing for
 Storyboard, 205
 property lists
 binary property lists, 348-349
 creating, 341-345
 editing, 338-341
 explained, 337
 Property List editor, 345-347
 specialized property lists, 349-350
 text property lists, 348
 types of, 337-338
live issues, troubleshooting, 118-120
llvm compiler, 52-54, 367
llvm-gcc compiler, 54

loading

- data into LeagueDocument.m file, 250-251
- dynamic loading, 51-52

localization, 285-286

- adding, 286
- Info.plist, 300-302, 312
- MainMenu.xib, 288-291
- testing, 287-288
- window XIBs, 291-300

localization keys (Info.plist), 314-315**Log view, 82****logic testing, 155-158**

- CSV reader tests, 159-166
- test data for, 158-159

logs, build, 403-404**LSApplicationCategoryType key (Info.plist), 316****LSArchitecturePriority key (Info.plist), 316****LSBackgroundOnly key (Info.plist), 317****LSEnvironment key (Info.plist), 317****LSExecutableArchitectures key (Info.plist), 317****LSFileQuarantineEnabled key (Info.plist), 318****LSFileQuarantineExcludedPathPatterns key (Info.plist), 318****LSGetAppDiedEvents key (Info.plist), 317****LSHasLocalizedDisplayName key (Info.plist), 315****LSMinimumSystemVersion key (Info.plist), 317****LSMinimumSystemVersionByArchitecture key (Info.plist), 317****LSMultipleInstancesProhibited key (Info.plist), 317****LSPrefersCarbon key (Info.plist), 319****LSRequiresiPhoneOS key (Info.plist), 320****LSRequiresNativeExecution key (Info.plist), 317****LSUIElement key (Info.plist), 318****LSUIPresentationMode key (Info.plist), 318****LSVisibleInClassic key (Info.plist), 319****M****.m file extension, 50****Mac OS X applications. See also Passer Rating application (Mac OS X)**

- application bundles, 309-311
- ARC (Automatic Reference Counting), 246-248
- bindings, 253
 - Game popover example, 269-273
 - Game table example, 268-269
 - Passer table example, 266-268
 - running, 260-263
 - Team table example, 260
- building, 332
- creating, 240-243

custom views, 275-276

- NSView subclass, creating, 276-279
- properties, 282-283
- view controller, 279-282

files, 241-242

frameworks

- adding to builds, 327
- debugging, 332-335
- explained, 325
- framework targets, 326
- installing, 327-329, 331
- location of, 330-331
- populating, 326-327

goals, 239-240

Info.plist. See Info.plist file

- LeagueDocument.m, loading data into, 250-251

localization, 285-286

- adding, 286
- Info.plist, 300-302, 312
- MainMenu.xib, 288-291
- strings in code, 302-306
- testing, 287-288
- window XIBs, 291-300

menus, adding, 248-250

model, porting from iOS, 243-246

object controllers, 258-260

popover windows, 269-273

project options, 240-241

property lists

- binary property lists, 348-349
- creating, 341-345
- editing, 338-341
- explained, 337
- Property List editor, 345-347
- specialized property lists, 349-350
- text property lists, 348
- types of, 337-338

views

- Autosizing, 255-257
- laying out, 263-264
- table view, 254-255

Mac OS X file templates

- C and C++, 496
- Cocoa, 496
- Core Data, 497
- resource templates, 497
- user interface, 497

Mac OS X project templates

- application plug-in templates, 491
- application templates, 489-490
- framework & library templates, 490-491
- system plug-in templates, 492

- MAC_OS_X_VERSION_ACTUAL build variable, 476
 - MAC_OS_X_VERSION_MAJOR build variable, 476
 - MAC_OS_X_VERSION_MINOR build variable, 476
 - MACH_O_TYPE build variable, 478
 - macosx-dev mailing list, 501
 - MACOSX_DEPLOYMENT_TARGET build variable, 482
 - MacroMates TextMate, 505
 - macros
 - DEBUG, 451
 - NS_BLOCK_ASSERTIONS, 450
 - mailing lists, 501
 - Main Menu template, 497
 - main.c file, passer-rating project, 26
 - main.m file, 95
 - MainHTML key (Info.plist), 324
 - MainMenu.xib file, localization, 288-291
 - makefiles, 389-390
 - Manage Schemes command (Product menu), 447
 - managed-object models, 99
 - attributes, 100-102
 - entities, 100
 - managed-object classes
 - creating, 105-106
 - extending, 106-108
 - relationships, adding, 102-105
 - managing schemes, 447-448
 - Mapping Model template, 495
 - Mark as Resolved command (Source Control menu), 70
 - Mark Heap option, 444
 - markup keywords, 380
 - Master-Detail Application template, 487
 - measurements
 - with analyzer, 193-195
 - Automatic Reference Counting (ARC), 195-196
 - memory, 182
 - allocations, 182-187
 - leaks, 187-189
 - zombies, 189-193
 - speed, 173-182
 - meetings, 503
 - memory, 182
 - management
 - Automated Reference Counting (ARC), 466-467
 - in Objective-C 2.0, 462-464
 - troubleshooting, 146, 148-149
 - measuring
 - allocations, 182-187
 - leaks, 187-189
 - zombies, 189-193
 - memory instruments, 426-427
 - Memory Monitor instrument, 428
 - menus
 - adding to Mac OS X applications, 248-250
 - help menu, 372
 - Merge command (Source Control menu), 84
 - merging, 76-77
 - message invocation in Objective-C, 456
 - method names, refactoring, 114
 - methods
 - configureCell:atIndexPath:, 116-117
 - dynamic dispatch in Objective-C 2.0, 470-471
 - fetchedResultsController, 117
 - fillWithData:, 250-251
 - names, refactoring, 114
 - passerEditorSaved:, 219
 - passerTableClicked: method, 281-282
 - persistentStoreCoordinator, 111
 - prepareForSegue:sender:, 215-216, 218
 - windowControllerDidLoadNib: method, 281
 - Mode panel (Doxygen), 383
 - Model-View-Controller (MVC) design pattern, 87-90
 - models, 88-89
 - attributes, 100-102
 - entities, 100
 - managed-object models, 99
 - porting from iOS to Mac OS X, 243-246
 - relationships, adding, 102-105
 - Modified file state, 68-69
 - mogenerator, 506
 - .mom files, 99
 - More Cocoa Programming for Mac OS X: The Big Nerd Ranch Guide* (Hillegass), 499
 - Multicore instrument template, 435
 - multiple editors (Xcode 4), 356-358
 - MVC (Model-View-Controller) design pattern, 87-90
- ## N
- NATIVE_ARCH build variable, 476
 - NATIVE_ARCH_32_BIT build variable, 476
 - NATIVE_ARCH_64_BIT build variable, 476
 - Network Activity instrument, 429
 - Network Activity Monitor instrument, 428
 - Network Connections instrument template, 434
 - Neuberg, Matt, 500
 - New command (File menu), 18

New File assistant sheet, 26
 New File command (New menu), 60
 New menu commands, New File, 60
 New Project command, 18
 New Project command (File menu), 90, 240
 nm tool, 55
 Not Controlled file state, 68
 NSAppleScriptEnabled key (Info.plist), 318
 NSArray data type, 468
 NSBundle class, 285
 NSCoder Night, 503
 NSData data type, 468
 NSDate data type, 468
 NSDictionary data type, 468
 NSFetchedResultsController class, 116
 NSHumanReadableCopyright key (Info.plist), 316
 NSJavaNeeded key (Info.plist), 320
 NSJavaPath key (Info.plist), 320
 NSJavaRoot key (Info.plist), 320
 NSMainNibFile key (Info.plist), 314
 NSManagedObject subclass template, 495
 NSNumber data type, 468
 NSPersistentStoreTypeKey key (Info.plist), 319
 NSPrefPanelconFile key (Info.plist), 323
 NSPrefPanelconLabel key (Info.plist), 323
 NSPrincipalClass key (Info.plist), 314
 NSServices key (Info.plist), 318
 NSSet data type, 469
 NSString data type, 468
 NSSupportsSuddenTermination key (Info.plist), 317
 NSValue data type, 469
 NSView class, creating subclasses, 276-279
 NS_BLOCK_ASSERTIONS macro, 450

O

objc-language mailing list, 501
 object controllers, Team Array Controller, 258-260
 OBJECT_FILE_DIR build variable, 479
 object files, 50, 55
 Object Graph instrument, 427
Objective-C
 class implementation, 458-460
 class interfaces, 457-458
 formal protocols, 456
 message invocation, 456
 need for, 455
Objective-C 2.0, Cocoa and, 460
 dynamic dispatch, 470-471
 fast enumeration, 467-468
 Foundation data types, 468-469
 Key-Value Coding (KVC), 461-462
 memory management, 462-464
 properties, 464-467
 version support, 461
Objective-C category template, 494, 496
Objective-C class template, 493, 496
Objective-C Programming: The Big Nerd Ranch Guide (Hillegass), 499
Objective-C protocol template, 494, 496
Objective-C test case class template, 494, 496
Objective-C++, 471
OBJROOT build variable, 478
 obsolete keys (Info.plist), 320
 Omelet property list, 341-345
Open As menu commands
 Property List, 346
 Source Code, 342, 345
Open command (File menu), 18
Open Quickly dialog, 371-372
OpenGL Driver instrument, 426
OpenGL ES Analysis instrument template, 434
OpenGL ES Analyzer instrument, 426
OpenGL ES Driver instrument, 426
OpenGL ES Driver instrument template, 434
OpenGL Game template, 487
Organization program (iOS developer program), 221-222
OS build variable, 475
OSAScriptingDefinition key (Info.plist), 318
OTHER_CFLAGS build variable, 481
 otool, 55
 outlets, 134-136
 Output panel (Doxygen), 383

P

packages. See also bundles
 benefits of, 307
 definition of, 307
 Installer package, 13-15
 RTFD (rich text file directory), 308-309
PACKAGE_TYPE build variable, 477
Page-Based Application template, 487
PassCompletionView
 PassCompletionView.h, 276-277
 PassCompletionView.m, 277-279
 PasserGraphController, 279-282
 properties, 282-283
passer-detail view
 checking connections, 137
 connecting GameListController, 137-139
 creating, 130-132
 labels, 132-134

- outlets, 134-136
- testing, 134, 141
- Passer editor**
 - editor controller, 212-215
 - editor view, 210, 212
- Passer list, reconstructing for Storyboard, 205**
- Passer Rating application (Mac OS X). See also Passer Rating project (iOS); passer-rating project (Mac command-line tool)**
 - ARC (Automatic Reference Counting), 246-248
 - bindings, 253
 - Game popover, 269-273
 - Game table, 268-269
 - Passer table, 266-268
 - running, 260-263
 - Team table, 260
 - building, 332
 - creating, 240-243
 - custom views, 275-276
 - NSView subclass, creating, 276-279
 - properties, 282-283
 - view controller, 279-282
 - files, 241-242
 - LeagueDocument.m, loading data into, 250-251
 - localization, 285-286
 - adding, 286
 - Info.plist, 300-302, 312
 - MainMenu.xib, 288-291
 - strings in code, 302-306
 - testing, 287-288
 - window XIBs, 291-300
 - menus, adding, 248-250
 - model, porting from iOS, 243-246
 - PasserGraph framework
 - adding to build, 327
 - debugging, 332-335
 - files, 326
 - framework target, 326
 - installing, 327-329, 331
 - location of, 330-331
 - populating, 326-327
 - popover windows, 269-273
 - project options, 240-241
 - property lists
 - binary property lists, 348-349
 - creating, 341-345
 - editing, 338-341
 - explained, 337
 - Property List editor, 345-347
 - specialized property lists, 349-350
 - text property lists, 348
 - types of, 337-338
 - Team Array Controller, 258-260
 - views
 - Autoresizing, 255-257
 - laying out, 263-264
 - table view, 254-255
- Passer Rating-Info.plist, 95**
- Passer Rating project (iOS), 221, 239. See also Passer Rating application (Mac OS X); passer-rating project (Mac command-line tool)**
 - adding files to, 97-98
 - Automatic Reference Counting (ARC), 195-196
 - classes
 - creating, 105-106
 - extending, 106-108
 - creating, 90-91
 - debugging, 111
 - GameListController, connecting, 137-139
 - generate-games.rb script, 108-111
 - memory, 182
 - allocations, 182-187
 - leaks, 187-189
 - zombies, 189-193
 - method names, refactoring, 114-115
 - model, 99
 - attributes, 100-102
 - entities, 100
 - relationships, 102-105
 - project template, 95-96
 - reconstructing for Storyboard, 201
 - custom table view cell, 207-210
 - model files, 203-204
 - Passer editor, 210, 212-215
 - Passer list, 205
 - segues, 215-219
 - views, 205-206
 - workspaces, 201-203
 - running, 123-124
 - searching/replacing fields, 93-94
 - snapshots, 94
 - speed, measuring, 173-182
 - target editor, 92-93
 - troubleshooting
 - with Debug navigator, 120-122
 - live issues and Fix-it, 118-120
 - view controllers, 116
 - adding, 127-128
 - fetchResultsController method, 117
 - Passers, creating, 117-118
 - table view, 116-117
 - XIB files, 128-129

- views
 - checking connections, 137
 - creating, 130-132
 - labels, 132-134
 - outlets, 134-136
 - testing, 134, 141
- passer-rating project (Mac command-line tool), 25. See also Passer Rating application (Mac OS X); Passer Rating project (iOS)**
 - build errors, 28-30
 - building, 28-30
 - creating, 25-28
 - debugging, 32-33, 35, 42
 - Behaviors panel, 40-42
 - breakpoints, setting, 36-37
 - stepping through code, 38-40
 - variables pane, 37-38
 - main.c, 26
 - passer-rating target
 - adding, 57-58
 - adding files to, 59-61
 - dependent targets, 62-63
 - headers in, 61
 - rating.c, 27-28
 - rating.h, 28
 - running, 30-32
- Passer table, binding, 266-268**
- passerEditorSaved: method, 219**
- PasserGraph framework**
 - adding to build, 327
 - debugging, 332-335
 - files, 326
 - framework target, 326
 - installing, 327-329, 331
 - location of, 330-331
 - populating, 326-327
- PasserGraph.h file, 326**
- PasserGraph-Info.plist file, 326**
- PasserGraph.m file, 326**
- PasserGraph-Prefix.pch file, 326**
- PasserGraphController, 279-282**
 - PasserGraphController.h, 279
 - PasserGraphController.m, 280
- PasserListController class, 114-115**
- Passers, creating, 117-118**
- passerTableClicked: method, 281-282**
- .pch file extension, 56**
- Perforce, 367**
- persistentStoreCoordinator method, 111**
- phases (build), 390-391**
- planning apps, Model-View-Controller (MVC)**
 - design pattern, 87-90
- Platforms directory, 11**
- PLATFORM_NAME build variable, 475**
- plists. See property lists**
- Plug-in key (Info.plist), 324**
- plug-ins**
 - application plug-in templates, 491
 - Info.plist keys, 322
 - system plug-in templates, 492
- popover windows, 269-273**
- populating frameworks, 326-327**
- PR-Storyboard project, 199-201**
- #pragma mark directive, 144**
- PRAppDelegate class, 95**
- PRDetailViewController class, 95**
- precompiled headers, 56, 407**
- predictive compilation, 54**
- Preference Pane template, 492**
- preferences**
 - Info.plist keys, 323
 - Key Bindings, 439
 - setting with defaults command-line tool, 437
- Preferences window, 21**
- prefix files, 56**
- prepareForSegue:sender: method, 215-218**
- preprocessing configuration files (.xcconfig), 399-400**
- preprocessors, 55**
- PRMasterViewController class, 95, 114**
- Process instrument, 428**
- Product menu commands**
 - Analyze, 54, 193-195
 - Attach to Process, 332
 - Build, 30, 118
 - Clean, 23
 - Generate Output, 55
 - Manage Schemes, 447
 - Run, 21, 28, 96
 - Stop, 33
- PRODUCT_NAME build variable, 477**
- profiles**
 - distribution profiles, 229-230
 - provisioning profiles, 223
 - sharing, 233-234
 - Team Provisioning Profile, 228
- Programming in Objective-C 2.0 (Kochan), 499**
- Programming iOS 4: Fundamentals of iPhone, iPad, and iPod Touch Development (Neuberg), 500**
- programs, compiling**
 - compiler products, 50, 55
 - dynamic loading, 51-52
 - explained, 45-49
 - gcc compiler, 52
 - linking, 50-51
 - llvm compiler, 52-54

- llvm-gcc compiler, 54
- precompiled headers, 56
- predictive compilation, 54
- PROJECT build variable, 475**
- Project Builder, 11**
- PROJECT_DIR build variable, 478**
- PROJECT_FILE_PATH build variable, 478**
- Project Info window, equivalents in Xcode 4, 361**
- PROJECT_NAME build variable, 477**
- Project navigator, 439**
- Project panel (Doxygen), 382-384**
- Project Search window, equivalent in Xcode 4, 362**
- project templates**
 - iOS projects
 - application templates, 487-488
 - framework & library templates, 488
 - Mac OS X projects
 - application plug-in templates, 491
 - application templates, 489-490
 - framework & library templates, 490-491
 - system plug-in templates, 492
- projects, 221, 390. See also apps**
 - App Store releases, preparing. See provisioning
 - Automatic Reference Counting (ARC), 195-196
 - debugging. See debugging
 - deleting, 23
 - Hello World project. See Hello World project
 - Info.plist. See Info.plist file
 - memory, 182
 - allocations, 182-187
 - leaks, 187-189
 - zombies, 189-193
 - Passer Rating. See Passer Rating project (iOS)
 - passer-rating. See passer-rating project (Mac command-line tool)
 - provisioning. See provisioning
 - speed, measuring, 173-182
 - version control. See version control
- properties**
 - custom view properties, 282-283
 - in Objective-C 2.0, 464-467
- Property List command (Open As menu), 346**
- Property List editor, 345-347**
- Property List template, 495**
- property lists**
 - creating, 341-345
 - editing, 338-341
 - explained, 337

- Property List editor, 345-347
- types of, 337-338
- protocols, formal protocols in Objective-C, 456**
- provisioning**
 - Automatic Device Provisioning (ADP), 223-225
 - distribution builds, 231-233
 - explained, 222-223
 - iOS Provisioning Portal, 225
 - application IDs, 227-228
 - Development Certificates, 225
 - device IDs, 226-227
 - Distribution Certificates, 225-226
 - distribution profiles, 229-230
 - Team Provisioning Profile, 228
 - preparing App Store releases, 234-236
 - provisioning profiles, 223
 - sharing identities and profiles, 233-234
 - signing identities, 230-231, 233-234
- provisioning profiles, 223**
- PRSPassEditController class, 212-215**
- Push command (Source Control menu), 73**
- pushing to remote repositories, 72-74**
- Python, bindings to Cocoa, 455**

Q

- QLNeedsToBeRunInMainThread key (Info.plist), 319**
- QLPreviewHeight key (Info.plist), 318**
- QLPreviewWidth key (Info.plist), 318**
- QLSupportsConcurrentRequests key (Info.plist), 318**
- QLThumbnailMinimumSize key (Info.plist), 318**
- Quartz Composer Plug-in template, 491**
- Quick Help for Selected Item command (Help menu), 370**
- Quick Help inspector, 369-370**
- Quick Help popover, 370-371**
- Quick Look Plug-in template, 492**

R

- rating.c file (passer-rating project), 27-28**
- rating.h file (passer-rating project), 28**
- Raymond, Eric, 500**
- Reads/Writes instrument, 426**
- Record Options command (File menu), 423**
- recording into trace documents, 421-422**
- Refactor command (Edit menu), 114**
- Refactor menu commands, Rename, 114**

refactoring

- class names, 114-115
- method names, 114

reference counting

- Automated Reference Counting (ARC), 466-467
- in Objective-C 2.0, 463

referencing embedded files in test classes, 166**Refresh Status command (Source Control menu), 68****relationships, adding, 102-105****Relative to Group option, 439****remote repositories, 71**

- pushing to, 72-74
- setting up, 71-72

REMOVE_CVS_FROM_RESOURCES build variable, 482**REMOVE_SVN_FROM_RESOURCES build variable, 482****removing**

- projects, 23
- Xcode, 12

Rename command (Refactor menu), 114**renaming**

- files, 440
- symbols, 114-115

reopening trace documents, 423**replacing fields (Passer Rating project), 93-94****repositories**

- cloning, 74-75
- Git repositories, 66-69
- remote repositories, 71-74
- Subversion repositories, 68

Request a Certificate from a Certificate

- Authority command (Keychain Access), 225

requirements for Xcode, 9-10**resizing with Autoresizing, 255-257****resource forks, 307****Resource Rules template, 496****resource templates**

- iOS file templates, 495-496
- Mac OS X file templates, 497

resources

- blogs, 502-503
- books, 499-500
- classes, 503
- Developer Technical Support (DTS), 501-502
- forums, 500-501
- mailing lists, 501
- meetings, 503
- resource forks, 307
- text editors, 504-505

tools, 505-506

websites, 502-503

RTF File template, 495**RTFD (rich text file directory), 308-309****Ruby, bindings to Cocoa, 455****Run command (Product menu), 21, 28, 96****Run Script editor, 109****running**

- bindings, 260-263
- Doxygen, 384-385
- Instruments, 412-413
- Passer Rating app, 123-124
- projects, 21-23, 30-32

S**Sampler instrument, 428****Save as Template command (File menu), 423****saving trace documents, 423****scanf function, 32, 50****Scheduled for Addition file state, 68****Scheduled for Deletion file state, 68****schemes, 147-149, 447-448****Screen Saver template, 492****Script Phase Info window, equivalents in Xcode 4, 362****scripts, generate-games.rb, 108-111****SDKROOT build variable, 478****Search Documentation for Selected Text command (Help menu), 371****Search navigator, 362****searching**

- documentation, 374-375
- fields, Passer Rating project, 93-94

segues, 215-219**SenTestingKit, 157****SenTestingKit assertions, 168**

- equality tests, 169
- exceptions, 169-170
- simple tests, 169

Separate by Thread display option, 417**setters, memory management, 463-464****setting environment variables, 147-149****Settings Bundle template, 495****settings for build variables, 474-475****setValue:forKey: method, 461****SHALLOW_BUNDLE build variable, 479****Shared Memory instrument, 427****sharing identities and profiles, 233-234****Shark, 366****Shell Script template, 496****Shipley, Wil, 503**

- Show live issues option (Preferences), 21
- Show Obj-C Only display option, 418
- Show Raw Values & Keys command (Editor menu), 350
- signatures, 223
- signing certificates, 223
- signing identities, 230-234
- simple test assertions, 169
- Simulate Document command (Editor menu), 255
- Single View Application template, 488
- slash (/), 67
- Sleep/Wake instrument, 429
- smart groups, 360
- SMAuthorizedClients key (Info.plist), 319
- SMPrivilegedExecutables key (Info.plist), 319
- snapshots, 94
- snippets, 144
- source code, 45
- Source Code command (Open As menu), 342, 345
- Source Control in Xcode 4, 364-365. *See also* version control
- Source Control menu commands
 - Commit, 70
 - Mark as Resolved, 70
 - Merge, 84
 - Push, 73
 - Refresh Status, 68
- source files, compiling, 408-409
- source location build variables, 478
- SOURCE_ROOT build variable, 478
- source trees, 483
- special-purpose editors (Xcode 4), 362-364
- specialized property lists, 349-350
- speed, measuring, 173-182
- Spin Monitor instrument, 428
- Spotlight Importer template, 492
- Stack Overflow, 501
- stack traces, 31
- Staged file state, 69
- Stanford University, Cocoa programming classes at, 503
- starting Xcode, 17-18
- STAssertThrowsSpecific assertion, 170
- STAssertThrowsSpecificNamed assertion, 170
- STAssertTrueNoThrow assertion, 170
- Step In command (Debug menu), 39
- Step Out command (Debug menu), 39
- Step Over command (Debug menu), 38
- stepping through code, 38-40
- STL C++ Library template, 491
- Stop command (Product menu), 33

- Storyboard
 - explained, 197-199
 - PR-Storyboard project, 199-201
 - reconstructing Passer Rating app for, 201
 - custom table view cell, 207-210
 - model files, 203-204
 - Passer editor, 210, 212-215
 - Passer list, 205
 - segues, 215-219
 - views, 205-206
 - workspaces, 201-203
 - segues, 215-219
 - workspaces, 201-203
- Storyboard template, 494
- strings
 - localization, 302-306
 - view strings, translating, 291
- STRINGS_FILE_OUTPUT_ENCODING build variable, 483
- Strings File template, 496
- strings in code, 302-306
- Style menu (instrument configuration), 420
- subclasses of UITableViewCell, creating, 153
- SubEthaEdit, 505
- submitting App Store releases, 235-236
- Subversion repositories, file states, 68-69
- Sudden Termination instrument, 425
- Sudden Termination instrument template, 435
- Summary tab (Passer Rating project), 92-93
- Switch Branch button, 83
- Symbol navigator (Xcode 4), 364
- symbols, 48
 - filtering in Xcode 4, 360
 - renaming
 - class names, 114-115
 - method names, 114
- SYMROOT build variable, 479
- Sync Schema template, 492
- system instruments, 427-429
- system plug-in templates, 492
- system requirements for Xcode, 9-10
- system tools, 11, 14
- System Usage instrument template, 434

T

- Tabbed Application template, 488
- table cells
 - adding, 143-146
 - creating custom, 149-153
- table view, 116-117, 254-255
 - custom table view cell, 207-210

table-view controllers, 143**tables**

- connecting to view controllers, 143-146
- Game table, binding, 268-269
- heapshot-analysis table, 444
- Passer table, binding, 266-268
- Team table, binding, 260

tags, 151, 449**TARGET_BUILD_DIR build variable, 479****Target editor, Build Phases tab, 49****Target Info window, equivalents in Xcode 4, 360****Target menu (Instruments), 414-415****TARGET_NAME build variable, 477****target templates, 493****targets, 390**

- adding, 57-58
- adding files to, 59-61
- in application testing, 167
- build phases, 58
- definition of, 58
- dependent targets, 62-63
- framework targets, 326
- headers in, 61
- for logic tests, 156-157
- in Xcode 4, 359

Team Provisioning Profile, 228**Team table, binding, 260****technical support, 501-502****templates**

- C++ Standard Template Library, 469
- File Template library, 497-498
- instantiating, 485-487, 498
- instrument templates, 433-435
- iOS file templates, 493
 - C and C++, 494-495
 - Cocoa Touch, 493-494
 - Core Data, 495
 - resource templates, 495-496
- iOS project templates
 - application templates, 487-488
 - framework & library templates, 488
- Mac OS X file templates
 - C and C++, 496
 - Cocoa, 496
 - Core Data, 497
 - resource templates, 497
 - user interface, 497
- Mac OS X project templates
 - application plug-in templates, 491
 - application templates, 489-490
 - framework & library templates, 490-491
 - system plug-in templates, 492
 - Passer Rating project, 95-96
 - target templates, 493

Terminal, 23**test classes, referencing embedded files, 166****test data for logic tests, 158-159*****Test-Driven iOS Development* (Graham), 500****testing**

- localization, 287-288
- views, 134, 141

testing frameworks, 155. See also unit testing**text editors, 504-505****text property lists, 348****TextMate, 505****TextWrangler, 504****Thread States instrument, 429****Time Profiler instrument, 428****Time Profiler instrument template, 434****tips. See tricks and techniques****toolbars, trace document window, 414-416****tools, 505-506. See also names of specific tools**

- Accessorizer, 505
- AppKiDo, 505
- AppKiDo-for-iPhone, 505
- mogenerator, 506
- otool, 55
- text editors, 504-505

touch tool, 410**trace documents, 413**

- Detail area, 417-418
- Extended Detail area, 418-419
- Library palette, 419
- recording into, 421-422
- saving and reopening, 423
- toolbar, 414-416
- Track area, 416-417

Track area (trace documents), 416-417**tracking memory allocations, 182-187****transcripts, build transcript example, 404-405**

- compiling source files, 408-409
- creating universal binaries, 410
- linking, 409
- precompiled header, 407
- resources, 406
- touch tool, 410

translating view strings, 291**traps, 448-451****tricks and techniques, 437**

- assistant editor, 441-442
- builds, 445-447
- Code Folding Ribbon, 440-441
- general suggestions, 437-440
- instruments and debugging, 443-445
- Interface Builder, 442
- jump bar, 440
- schemes, 447-448

troubleshooting. See debugging**Type menu (instrument configuration), 421**

U

- UI automation instruments, 429
- UI Recorder instrument template, 435
- UIAppFonts key (Info.plist), 320
- UIApplicationExitsOnSuspend key (Info.plist), 322
- UIBackgroundModes key (Info.plist), 322
- UIDeviceFamily key (Info.plist), 320
- UIFileSharingEnabled key (Info.plist), 322
- UIInterfaceOrientation key (Info.plist), 321
- UILabel class, 132
- UILaunchImageFile key (Info.plist), 321
- UIMainStoryboardFile key (Info.plist), 314
- UIPrerenderedIcon key (Info.plist), 321
- UIRequiredDeviceCapabilities key (Info.plist), 320
- UIRequiresPersistentWiFi key (Info.plist), 320
- UIStatusBarHidden key (Info.plist), 321
- UIStatusBarStyle key (Info.plist), 321
- UINavigationController class, 200
- UISupportedExternalAccessoryProtocols key (Info.plist), 321
- UISupportedInterfaceOrientations key (Info.plist), 321
- UITableViewCell, creating subclasses, 153
- UIUpgradeOtherBundleIdentifier key (Info.plist), 319
- UIViewController class, 90
- UIViewEdgeAntialiasing key (Info.plist), 321
- UIViewGroupOpacity key (Info.plist), 321
- uninstall-developer-folder file, 12
- uninstall-devtools file, 12
- uninstalling. *See* removing
- unit testing, 155
 - application tests, 166-168
 - logic tests, 156-158
 - CSV reader testing, 159-166
 - test data for, 158-159
 - SenTestingKit assertions, 168
 - equality tests, 169
 - exceptions, 169-170
 - simple tests, 169
- universal binaries, creating, 410
- UNIX development suite, 11, 14
- Unknown file state, 70
- UNLOCALIZED_RESOURCES_FOLDER_PATH
 - build variable, 480
- Unmerged file state, 69
- Unmodified file state, 68-69
- Untracked file state, 69
- URLs, Info.plist keys, 314-315
- USER build variable, 475
- user information keys (Info.plist), 313-314, 316

- user interface, Mac OS X file templates, 497
- User Interface instrument, 426
- user interface instruments, 430
- user presentation keys (Info.plist), 321-322
- /usr/bin directory, 11
- UTExportedTypeDeclarations key (Info.plist), 315, 349
- Utility Application template, 488
- Utility area (Xcode 4), 365
- UTImportedTypeDeclarations key (Info.plist), 315

V

- VALID_ARCHS build variable, 480
- validating App Store releases, 235-236
- valueForKey: method, 461
- variables, build
 - editing, 395-396
 - explained, 392-393
 - settings hierarchy, 393-395
- variables pane, 37-38
- version control
 - Blame view, 81-82
 - branching, 82-84
 - cloning, 74-75
 - commits, 70-71
 - Comparison view, 79-81
 - conflicts, 75-79
 - explained, 65-66
 - file states
 - in Git, 68-69
 - in Subversion, 68
 - in Xcode, 69-70
 - Git, 26
 - Git repositories, creating by hand, 66-68
 - Log view, 82
 - merging, 76-77
 - remote repositories, 71
 - pushing to, 72-74
 - setting up, 71-72
- Version editor
 - Blame view, 81-82
 - Comparison view, 79-81
 - Log view, 82
- vertical spacing (labels), 297
- vi, 504
- view controllers, 90, 113
 - adding, 127-128
 - connecting to tables, 143-146
 - editing, 116-117
 - GameListController, connecting, 137-139
 - PasserGraphController, 279-282

Passers, creating, 117-118
XIB files, 128-129

View menu commands, Assistant Editor, 356

view strings, translating, 291

View template, 494, 497

views, 89-90

- Autoresizing, 255-257
- Blame view, 81-82
- checking connections, 137
- Comparison view, 79-81
- connecting GameListController, 137-139
- creating, 130-132
- custom views (Mac OS X), 275-276
 - NSView subclass, creating, 276-279
 - properties, 282-283
 - view controller, 279-282
- labels, 132-134
- laying out, 263-264
- Log view, 82
- outlets, 134-136
- passer-detail view
 - checking connections, 137
 - connecting GameListController, 137-139
 - creating, 130-132
 - labels, 132-134
 - outlets, 134-136
 - testing, 134, 141
- table view, 116-117, 254-255
 - custom table view cell, 207-210
- testing, 134, 141
- view controllers. *See* view controllers

VM Tracker instrument, 427

W

WARNING_FLAGS build variable, 481

warnings, definition of, 29

websites, 502-503

Welcome to Xcode window, 18

widgets, Info.plist keys, 323-324

Width key (Info.plist), 323

WiFi instrument, 429

Window template, 495, 497

window XIBs, localization

- autolayout, 292-300

- translating view strings, 291

viewControllerDidLoadNib: method, 281

Workspace (Xcode 4), 354

Workspace window

- closing, 23

- Debug area, 22-23

workspaces, 201-203

WRAPPER_NAME build variable, 477

X-Z

.xcconfig files, 398

- creating, 398

- preprocessing, 399-400

- SDK- and architecture-specific settings, 399

.xcdatamodel file, 99

Xcode 4, 353

- behaviors, 362-363

- browsers, 353-355, 364

- build process, 358

- desktop, 353-355

- editors

- assistant editor, 355-356

- Editor control, 355

- multiple editors, 356-358

- filtering files/symbols in, 360

- groups and files, 358-360

- Interface Builder, 365-366

- miscellaneous changes in, 366-367

- Source Control, 364-365

- special-purpose editors, 362-364

- Utility area, 365

- Workspace window, 354

- Xcode 3 Info windows, equivalents in, 360-362

Xcode how-to's, 373

xcode-users mailing list, 501

XCODER_VERSION_ACTUAL build variable, 477

XCODER_VERSION_MINOR build variable, 477

XCODER_VERSION_MAJOR build variable, 477

xcodebuild tool, 400-401

XIB files, 96, 128-129

- localization

- autolayout, 292-300

- translating view strings, 291

XPC Service template, 491

zombies, 149, 189-193

Zombies instrument template, 434

Zoom menu (instrument configuration), 421