Mastering Xcode 4

DEVELOP AND DESIGN

Joshua Nozzi

Mastering Xcode 4

DEVELOP AND DESIGN

Joshua Nozzi

Peachpit Press

Mastering Xcode 4: Develop and Design
Joshua Nozzi

Peachpit Press
1249 Eighth Street
Berkeley, CA 94710
510/524-2178
510/524-2221 (fax)

Find us on the Web at: www.peachpit.com
To report errors, please send a note to errata@peachpit.com
Peachpit Press is a division of Pearson Education
Copyright © 2012 by Joshua Nozzi

Editor: Cliff Colby
Production editor: Myrna Vladic
Development editor: Kim Wimpsett and Robyn G. Thomas
Copyeditor: Scout Festa
Technical Editor: Duncan Campbell
Cover design: Aren Howell Straiger
Interior design: Mimi Heft
Compositor: David Van Ness
Indexer: Ann Rogers

Notice of Rights
All rights reserved. No part of this book may be reproduced or transmitted in any form by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher. For information on getting permission for reprints and excerpts, contact permissions@peachpit.com.

Notice of Liability
The information in this book is distributed on an "As Is" basis, without warranty. While every precaution has been taken in the preparation of the book, neither the author nor Peachpit Press shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the instructions contained in this book or by the computer software and hardware products described in it.

Trademarks
Xcode is a trademark of Apple Inc., registered in the United States and other countries. Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and Peachpit Press was aware of a trademark claim, the designations appear as requested by the owner of the trademark. All other product names and services identified throughout this book are used in editorial fashion only and for the benefit of such companies with no intention of infringement of the trademark. No such use, or the use of any trade name, is intended to convey endorsement or other affiliation with this book.

ISBN 13: 978-0-321-76752-3
ISBN 10: 0-321-76752-7

9 8 7 6 5 4 3 2 1

Printed and bound in the United States of America

Thanks to all my peers, friends, and family for their enthusiastic support,

to a great team of professionals for helping me reach this goal,

and to Matt for putting up with yet another of

my time-consuming projects.

ACKNOWLEDGMENTS

I wish to thank the following people whose work I used while writing this book.

CYRIL GODEFROY

Cyril's masterfully broken code examples demonstrated some nice highlights of the Clang Static Analyzer. You can find them at *http://xcodebook.com/cgodefroy*.

COLIN WHEELER

Colin's Xcode shortcut cheat sheet saved me loads of tedium when creating Appendix B. You can find the original, downloadable version that Colin maintains at *http://xcodebook.com/cwheeler*.

CONTENTS

Introduction x
Welcome to Xcode xii

PART I THE BASICS: GETTING STARTED WITH XCODE 4

CHAPTER 1 **DISCOVERING XCODE TOOLS** **2**
Downloading 4
Installing 4
Exploring 5
Wrapping Up 7

CHAPTER 2 **STARTING A PROJECT** 8
Welcome to Xcode! 10
Creating a New Project 11
Project Modernization 15
Building and Running an Application 17
Wrapping Up 17

CHAPTER 3 **NAVIGATING A PROJECT** 18
The Workspace Window 20
The Navigator Area 21
The Jump Bar 26
The Editor Area 27
The Utility Area 31
The Debug Area 32
The Activity Viewer 33
The Tabs 34
The Organizer Window 35
Wrapping Up 35

CHAPTER 4 **GETTING HELP** **36**
The Help Menu 38
The Organizer's Documentation Tab 39
The Source Editor 41
Community Help and Feedback 42
Wrapping Up 43

PART II WORKING WITH COCOA APPLICATIONS

CHAPTER 5 CREATING USER INTERFACES . . . 46
Understanding Nibs . . . 48
Getting Familiar with Interface Builder . . . 50
Adding User Interface Elements . . . 55
Layout . . . 58
Wrapping Up . . . 69

CHAPTER 6 ADDING FILES TO A PROJECT . . . 70
Adding Existing Files . . . 72
Creating New Files . . . 74
Using the File Template Library . . . 76
Removing Files from the Project . . . 77
Wrapping Up . . . 77

CHAPTER 7 WRITING CODE WITH THE SOURCE EDITOR . . . 78
Exploring the Source Editor Interface . . . 80
Navigating Source Code . . . 81
Using Code Completion . . . 84
Exploring the Code Snippet Library . . . 85
The Assistant . . . 87
Wrapping Up . . . 87

CHAPTER 8 SEARCHING AND REPLACING . . . 88
Using the Search Navigator . . . 90
Searching within Files . . . 97
Wrapping Up . . . 97

CHAPTER 9 BASIC DEBUGGING AND ANALYSIS . . . 98
Compile-Time Debugging . . . 100
Runtime Debugging . . . 102
Wrapping Up . . . 109

CHAPTER 10 USING THE DATA MODEL EDITOR . . . 110
Introducing Core Data . . . 112
Using the Data Model Editor . . . 115

Creating a Basic Data Model 117
Creating a UI for the Model 118
Using the Assistant 125
Wrapping Up 125

CHAPTER 11 **CUSTOMIZING THE APPLICATION ICON** **126**
Picking the Ideal Artwork 128
Creating Icons 129
Setting the Application Icon 131
Setting Document Icons 133
Wrapping Up 133

CHAPTER 12 **DEPLOYING AN APPLICATION** **134**
Archiving 136
Alternatives to Archiving 140
Wrapping Up 141

PART III GOING BEYOND THE BASICS

CHAPTER 13 **ADVANCED EDITING** **144**
Renaming Symbols 146
Refactoring 147
Organizing with Macros 150
Changing Editor Key Bindings 151
Jump to Definition 152
My Company Name 153
Wrapping Up 153

CHAPTER 14 **THE BUILD SYSTEM** **154**
An Overview 156
Working with Targets 159
Working with Schemes 178
Entitlements (Sandboxing) 191
Wrapping Up 193

CHAPTER 15 **LIBRARIES, FRAMEWORKS, AND LOADABLE BUNDLES** **194**
What are Libraries, Frameworks, and Bundles? 196
Using Existing Libraries and Frameworks 199
Creating a Framework 208
Wrapping Up 215

CHAPTER 16 **WORKSPACES** **216**
What Is a Workspace? 218
When to Use a Workspace 220
Creating a Workspace 221
Another Kind of Workspace 228
Wrapping Up 231

CHAPTER 17 **DEBUGGING AND ANALYSIS IN DEPTH** **232**
Using the Clang Static Analyzer 234
Exploring Analyzer Results 236
Threads and Stacks 242
Inspecting Memory 246
Conferring with the Console 250
Viewing Generated Output 258
Debugging Apps for iOS Devices 260
Wrapping Up 269

CHAPTER 18 **UNIT TESTING** **270**
What is Unit Testing? 272
Unit Testing in Xcode 276
Writing a Unit Test 284
Adding Unit Tests to Existing Projects 295
Wrapping Up 297

CHAPTER 19 **USING SCRIPTING AND PREPROCESSING** **298**
Extending Your Workflow with Custom Scripts 300
Using the Preprocessor 313
Wrapping Up 322

CHAPTER 20 **USING INSTRUMENTS** **324**

An Overview of DTrace 326

A Tour of Instruments 327

Using Instruments for Common Tasks 339

Wrapping Up 348

CHAPTER 21 **SOURCE CODE MANAGEMENT** **350**

Xcode Snapshots 352

Using an SCM System 356

Wrapping Up 373

Index 374

APPENDIXES

APPENDIX A **MANAGING YOUR iOS DEVICES** **A-1**

APPENDIX B **GESTURES AND KEYBOARD SHORTCUTS** **B-17**

APPENDIX C **DOCUMENTATION UPDATES** **C-32**

APPENDIX D **OTHER RESOURCES** **D-36**

INTRODUCTION

This book is an intermediate-level introduction to Xcode 4, Apple's integrated development environment. It assumes you have some development experience and are familiar with the Cocoa API. It won't teach you how to write code or much at all about Cocoa. There are other books for that. This one is strictly focused on how to use Xcode itself, whatever your development endeavors.

Of course, since Xcode is most often used with the Cocoa API and Objective-C, there are basic introductions to Cocoa concepts and a few trivial code samples sprinkled here and there to illustrate various points. In these cases, I point to the documentation that Apple provides (to save you some trouble looking it up), but I only had a limited number of pages in which to show you Xcode stuff, so please keep this in mind when writing your scathing Amazon reviews.

Also, I've formed the opinion that Apple is crafty when it comes to software releases. Not only are they ultra-secretive, but they appear to know my precise schedule and plans (I blame iCloud). They seem to use this knowledge to wait until I'm almost finished and then change a bunch of stuff in a single release, necessitating the tracking down and editing of many fine details. I imagine an Apple overseer watching me through my Mac's camera, stroking a wrinkly, hairless cat and waiting until I'm almost finished. He then orders his henchmen to release the next set of random changes and leans toward the screen expectantly, muttering "Yeeesssss . . ." as I shake my fist at the sky and shout his name in dramatic fashion. The cat, of course, is hairless to avoid messing up his black turtleneck.

Whatever the case, I may say things that no longer apply to some future version or mention menus that no longer exist as such. Sorry. Blame Apple. Then buy my next edition.

WHAT YOU WILL LEARN

This book is divided into three major parts and includes four appendixes on the book's companion Web site.

Part I: The Basics: Getting Started with Xcode 4
In very short order, you'll install Xcode and get down to business building a useless application. Nobody but perhaps your mother would buy it, but it very neatly demonstrates the Xcode 4 project workflow and how to find your way around a project.

Part II: Working with Cocoa Applications
Next, you'll learn how to build and edit user interfaces, add resources, and customize the application. You'll explore all major aspects of the Xcode user interface and its primary editors. You'll learn to refactor code, to use the debugger and the Core Data modeler, and to archive builds for deployment (independently or via the App Store).

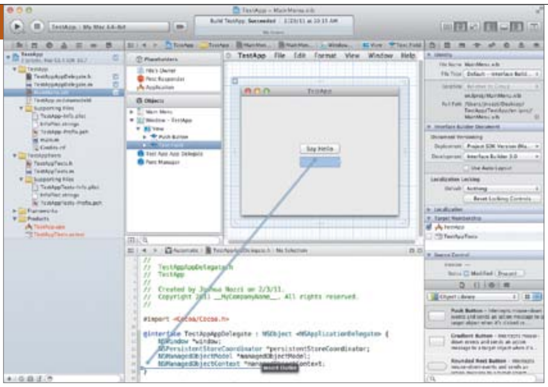
Part III: Going Beyond the Basics
Then you'll dive a little deeper and explore Xcode's build system (including the new schemes system). You'll learn how to create and use libraries and frameworks and how to combine multiple projects into a single workspace. You'll create and run unit tests and use custom scripts with the build process.

Finally, you'll take a solid tour of Instruments (Apple's profiling tool) and experience its uncanny ability to point out your mistakes and make you feel stupid. Thoroughly abashed, you'll wrap up with an overview of Xcode's integrated source code management support.

Appendixes
You'll find four appendixes on the book's companion Web site (*http://xcodebook.com/extracontent*). Appendix A helps you manage your iOS devices. Appendix B includes tables of gestures and keyboard shortcuts for frequently used tasks. Appendix C shows you how to manage Xcode documentation updates. Appendix D provides you with Apple and third-party resources for additional information.

WELCOME TO XCODE

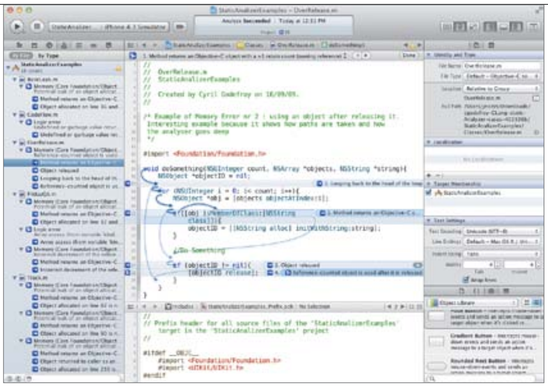
Upstart newbies. Always strolling in and making short work of stuff that used to take you hours. In your day, you *typed* all your build commands and *liked* it. Uphill. Both ways. In the snow. Then again, why let those newbies outpace you? Xcode puts the same powerful tools you know (and some new ones you may not) in your hands. Despite its shiny, easy-to-use interface, a lot of power lurks just under the



```
//
//  TestAppAppDelegate.h
//  TestApp
//
//  Created by Joshua Nozzi on 2/3/11.
//  Copyright 2011 __MyCompanyName__. All rights reserved.
//

#import <Cocoa/Cocoa.h>

@interface TestAppAppDelegate : NSObject <NSApplicationDelegate> {
    NSWindow *window;
    NSPersistentStoreCoordinator *persistentStoreCoordinator;
    NSManagedObjectModel *managedObjectModel;
    NSManagedObjectContext *managedObjectContext;
}
```



```
//
//  OverRelease.m
//  StaticAnalizerExamples
//
//  Created by Cyril Godefroy on 18/09/09.
//

/* Example of Memory Error nr 2 : using an object after releasing it.
 Interesting example because it shows how paths are taken and how
 the analyzer goes deep
 */

#import <Foundation/Foundation.h>

void doSomething(NSUInteger count, NSArray *objects, NSString *string){
    NSObject *objectID = nil;

    for (NSUInteger i = 0; i< count; i++){
        NSObject *obj = [objects objectAtIndex:i];

        if([[obj] isMemberOfClass:[NSString
            class]]){
            objectID = [[NSString alloc] initWithString:string];
        }

        //Do Something

        if (objectID != nil){
            [objectID release];
        }
    }
}
```

```
//
// Prefix header for all source files of the 'StaticAnalizerExamples'
   target in the 'StaticAnalizerExamples' project
//

#ifdef __OBJC__
    #import <Foundation/Foundation.h>
    #import <UIKit/UIKit.h>
#endif
```

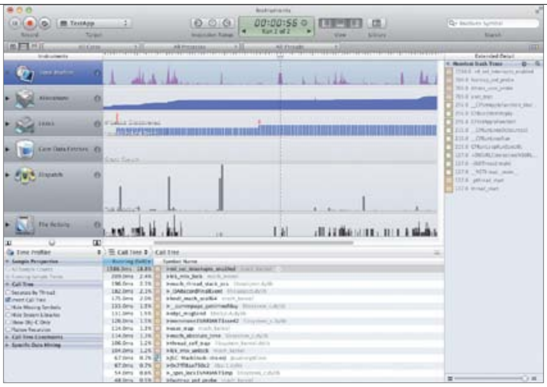
INTERFACE BUILDER

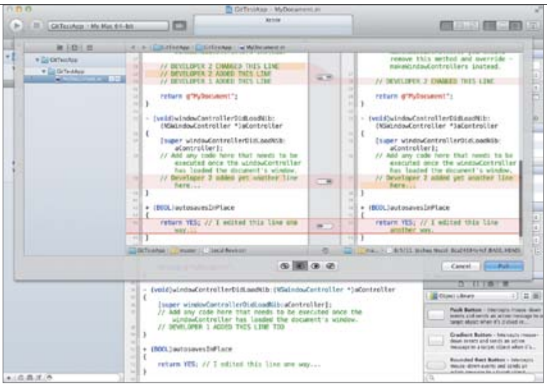
Build and edit rich user interfaces with Interface Builder. Drag and drop outlets and actions directly into your code using the Assistant editor.

CLANG STATIC ANALYZER

Find subtle errors in your programs with the Clang Static Analyzer. Follow the blue arrows through your code as the problem is broken down step by step.

surface. Xcode 4 lets you write and manage your code, design and build user interfaces, analyze and debug your apps, and more. So what if it takes you less time?





INSTRUMENTS

Trace and profile your code with Instruments. Follow your application's activity through time to find and analyze performance problems and more.

SOURCE CODE MANAGEMENT

Manage your source code with the integrated source code management features. Branch, merge, pull, push, and resolve conflicts all from within Xcode.

This page intentionally left blank

4

GETTING **HELP**

You can get help in Xcode 4 for the IDE as well as the Cocoa frameworks in a number of ways. In this chapter, you'll learn how to find the answers you need.

THE **HELP** MENU

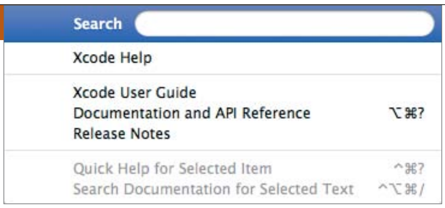

Search
Xcode Help
Xcode User Guide
Documentation and API Reference ⌥⌘?
Release Notes
Quick Help for Selected Item ^⌘?
Search Documentation for Selected Text ^⌥⌘/

FIGURE 4.1 The Xcode Help menu

You'll start with the most obvious place first. If you're familiar with Mac OS X, you should be familiar with the Help menu. Starting with Mac OS X 10.5, the Help menu (**Figure 4.1**) features a standard Search field, which shows you not only help topics but main menu items that match your search term.

Beneath the Search field are menu choices to open some of the more important help libraries with which you should familiarize yourself. A link to Xcode's release notes is also listed there. Most of these will open the Organizer window, which you saw in Chapter 3.

XCODE HELP

The Xcode User Guide menu item opens a splash page containing video and links about how to find help in Xcode.

XCODE USER GUIDE

The Xcode Help menu item opens the user manual for Xcode. Inside you'll find in-depth explanations and how-to instructions for all Xcode features.

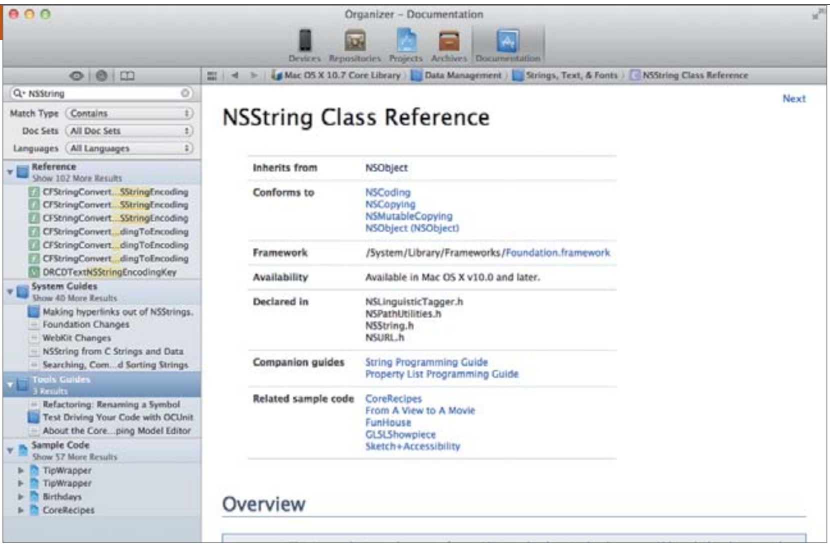
DOCUMENTATION AND API REFERENCE

The Documentation and API Reference menu item simply opens the Organizer window in Documentation mode so you can browse or search the installed documentation libraries. It will remain on the currently selected page without navigating away and simply show the window.

THE REST

The remaining menu items trigger quick help for the currently selected code in the active workspace window, and open the Organizer window in Documentation mode with the text input cursor in the search bar, respectively.

THE ORGANIZER'S **DOCUMENTATION** TAB


Organizer – Documentation
Devices
Repositories
Projects
Archives
Documentation
Mac OS X 10.7 Core Library
Data Management
Strings, Text, & Fonts
NSString Class Reference
NSString
Match Type
Contains
Doc Sets
All Doc Sets
Languages
All Languages
Reference
Show 102 More Results
CFStringConvert...SStringEncoding
CFStringConvert...SStringEncoding
CFStringConvert...SStringEncoding
CFStringConvert...dingToEncoding
CFStringConvert...dingToEncoding
CFStringConvert...dingToEncoding
DRCDTextNSStringEncodingKey
System Guides
Show 40 More Results
Making hyperlinks out of NSStrings.
Foundation Changes
WebKit Changes
NSString from C Strings and Data
Searching, Com...d Sorting Strings
Tools Guides
3 Results
Refactoring: Renaming a Symbol
Test Driving Your Code with OCUnit
About the Core...ping Model Editor
Sample Code
Show 57 More Results
TipWrapper
TipWrapper
Birthdays
CoreRecipes
Next

NSString Class Reference

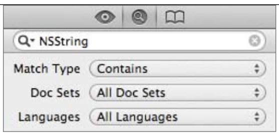
Inherits from	NSObject
Conforms to	NSCoding NSCopying NSMutableCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/Foundation.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	NSLinguisticTagger.h NSPathUtilities.h NSString.h NSURL.h
Companion guides	String Programming Guide Property List Programming Guide
Related sample code	CoreRecipes From A View to A Movie FunHouse GLSLShowpiece Sketch+Accessibility

Overview

FIGURE 4.2 The Organizer's Documentation tab

The Documentation section of the Organizer is Xcode's main viewer for all SDK and developer tools documentation. Open the Organizer by choosing Window > Organizer from the main menu, and then choose the Documentation tab in the toolbar (**Figure 4.2**).

The Organizer responds by showing you a pane on the left (similar to Xcode's Navigator area) and a main viewing area. Along the top of the main viewing area, you'll see a Jump Bar similar to the one you explored in Chapter 3. This Jump Bar, however, allows you to navigate the documentation as opposed to your project. The Organizer's navigation area has three modes: Explore, Search, and Bookmarks. The button bar at the top of the pane switches between the modes.



NSString

Match Type: Contains

Doc Sets: All Doc Sets

Languages: All Languages

FIGURE 4.3 The Organizer's search options panel

EXPLORE

In Explore mode, an outline of each of the documentation sets and their sections displays. You can drill down by topic through the guides and API reference documents.

SEARCH

In Search mode, a search field appears, allowing you to search all installed documentation sets. Clicking the magnifying glass icon, then choosing Show Find Options from the context menu reveals a set of filtering options (**Figure 4.3**) that let you ignore unwanted libraries and more. The results are grouped by type (such as Reference, System Guides, Sample Code, and so on).

BOOKMARKS

In Bookmarks mode, you can jump directly to documentation pages you've bookmarked. You can set bookmarks by choosing Editor > Bookmarks from the main menu or by right-clicking anywhere in the page and choosing Add Bookmark for Current Page from the context menu. To delete a bookmark, select it and press the Delete key.

THE **SOURCE** EDITOR

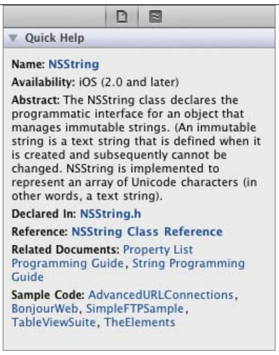
Although you'll explore the depths of the Source Editor in Part II, there are a couple of useful ways to find contextual help from within your code.

QUICK HELP IN THE UTILITY AREA

As you saw in Chapter 3, Xcode's Utility area gives you access to various properties, code snippets, user interface elements for Interface Builder, and Quick Help. The Quick Help utility continuously updates its content depending on what you've selected in the Source Editor.

To get a feel for this utility, make sure your TestApp project is open and then select the `TestAppAppDelegate.m` source file from the Classes group in the Project navigator. Open the Utility area, and select the Quick Help utility. In the Source Editor, click `window` in the `@synthesize window;` statement. Quick Help responds by showing you the name of the symbol (`window`) and the header file in which it is declared (the TestApp project's `TestAppAppDelegate.h` file).

For a more interesting example, scroll down a bit until you find the `- (NSString *) applicationSupportDirectory` method, and click the `NSString` symbol. Quick Help responds by showing a much more detailed description of the `NSString` class (**Figure 4.4**), which is part of the Cocoa frameworks and is documented by the built-in document libraries. Any text highlighted in blue is a hyperlink to the corresponding documentation. Clicking a Quick Help hyperlink opens the linked information in the Documentation section of the Organizer.

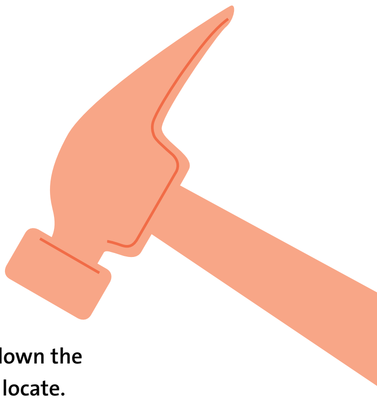


Name: NSString
Availability: iOS (2.0 and later)
Abstract: The NSString class declares the programmatic interface for an object that manages immutable strings. (An immutable string is a text string that is defined when it is created and subsequently cannot be changed. NSString is implemented to represent an array of Unicode characters (in other words, a text string).
Declared In: NSString.h
Reference: NSString Class Reference
Related Documents: Property List Programming Guide, String Programming Guide
Sample Code: AdvancedURLConnections, BonjourWeb, SimpleFTPSample, TableViewSuite, TheElements

FIGURE 4.4 The Quick Help utility

SEARCH DOCUMENTATION FOR SELECTED TEXT

Another handy way to find the documentation for symbols such as `NSString` is to select the symbol in the Source Editor and then choose Help > Search Documentation for Selected Text from the main menu. As with hyperlinks in the Quick Help pane, choosing this option will open any corresponding documentation it finds in the Documentation section of the Organizer.



TIP: It's not necessary to open the Utility pane to see Quick Help content. The same information will appear in a pop-up bubble by holding down the Option key and clicking the symbol you want to locate.

COMMUNITY **HELP** AND **FEEDBACK**

There are a number of community Web sites for finding more help than is available in the documentation, including Apple's own developer forums. See Appendix D for more information.

APPLE'S DEVELOPER FORUMS

Apple's developer forums are accessible to ADC members (*http://devforums.apple.com*). There you can receive help and advice from the Cocoa developer community as well as the occasional Apple engineer. Because this is a public forum, it's important to keep in mind that most people there are developers like you and are volunteering their time. Take extra care to search for similar questions before posting, ask detailed and clearly written questions, and be civil. As with any community, anything less than civility and courtesy will make the community less likely to help you in the future.

DOCUMENTATION ERRORS

If there is anything about Apple's documentation that is unclear, incorrect, or lacking in any way in Xcode, you are encouraged to submit feedback to Apple. At the bottom of every page of the documentation are hyperlinks that allow you to submit feedback—good, sort-of-good, and bad—to the Apple documentation team. Constructive, detailed feedback helps Apple provide better documentation, and improvements are released often.

WRAPPING UP

Xcode offers many ways to find help. Most of those ways point to the same documentation set, but even the documentation pages help you submit suggestions for improvement. In addition, you will find a number of community Web sites (one of which is Apple-hosted) and blogs a quick Google search away. Even if the built-in documentation doesn't help, the chances are quite good you'll find your answer on the World Wide Web.

You should now have your bearings in the Xcode IDE. In Part II, you'll use Xcode to expand on the TestApp project by adding some user interface elements, writing some code, defining a data model, and more. From there, you'll delve into the debugger and building the application for deployment.

This page intentionally left blank

INDEX

A

.a extension for libraries, 196
accessors, creating, 124
action buttons, adding, 55–56
actions
  adding and removing, 301
  nibs, 48
  post-Build-action actions, 301
Activity Viewer, 33
Add Files sheet, 72–73, 204
Add to Targets option, for adding files, 73
adding
  actions, 301
  breakpoints, 24
  files to projects. *See* files, adding to projects
  repositories manually, 359–361
addresses (memory), 246–247
analysis. *See* debugging and analysis
Analyze action, 189, 234
App Store, submitting applications to, 139
AppKit.framework, 197
Apple
  Cocoa documentation, 49
  developer forums, 42
  documentation errors, 42
  iOS Developer Program, 260
Application Help Books, 93
application icons, customizing, 127–133
  application icons, setting, 131–132
  artwork, 128
  document icons, setting, 133
  icons, creating, 129–130
applications. *See also* iOS devices, debugging apps for
  building and running, 17
  document-based/plain applications, 13
  naming, 12–13
  when to use workspaces, 220
applications, deploying, 135–141
  alternatives to archiving, 140
  archiving, 136–139
  Release mode and, 135–136
apps. *See* applications; iOS devices, debugging apps for
Apps controls (build systems), 193
ARCHIVE_DSYMS_PATH, 140
ARCHIVE_PATH, 140
ARCHIVE_PRODUCTS_PATH, 140
archiving, 136–140
  alternative to, 140
  applications, sharing, 139
  applications, submitting to App Store, 139
  applications, testing, 139
  Archive action (schemes), 189
  archives, creating, 136
  archives, examining, 138
  archives, finding, 137
  scripting example for (amended), 310–311
  scripting example for (simple), 307–309
Arguments tab
  Profile action, 188
  Run action, 185
  Test action, 186, 187
array controllers
  adding to UI for data model, 121
  binding columns to, 122
artwork
  Icon Composer and, 6, 129
  ideal, selecting for icons, 128
assertions, testing and, 279–280
Assistant
  data models and, 125
  in Editor area, 28–29
  Interface Builder, 50, 54
  viewing generated output and, 258–259
  writing code and, 80, 87
Assistant windows, Jump Bar in, 26
auto-creating schemes, 180, 181
Autolayout, 63–69
  activating, 64
  basics, 63
  constraints, 64–67, 68, 69
  new functionality, 68–69
automated refactoring, 147–148
Automatic behavior (Assistant), 54, 125
Autosizing control, 59, 60, 61, 62

B

backtrace, viewing, 254
badges (status), 364
bar pointer, memory addresses and, 246, 247
bash shell, 303
behaviors
  behavior modes (Assistant), 30, 87, 125
  Behaviors menu, 229
  Behaviors panel, 228
  when creating workspaces, 228–230
beta versions, 315–321
beta warnings, 316, 319, 321
bezel alert, creating, 230
bindings, Cocoa Bindings controller layer, 121, 122
/bin/sh shell, 301, 303, 306
Blame mode (Version Editor), 372
bookmarks, documentation pages and, 40
branches (Git), creating and switching, 369–370
Breakpoint navigator
  basics, 24
  runtime debugging and, 107
breakpoints
  runtime debugging and, 106–107
  vs. threads, 245
bt command, 254–255
Build action (schemes), 183–184

build environments, defining macros in, 316–321
build errors example, 101
Build Phases tab, 169–172, 206
Build Rules tab, 173–174
Build Settings sheet, 16
Build Settings tab, 165–168
build systems, 155–193
  build phases, 158, 169–172
  build rules, 158
  Build settings, 157
  configurations, 157
  entitlements, 191–193
  overview, 156–158
  run destinations, 157
  schemes. *See* schemes
  targets. *See* targets
build time vs. runtime, 322
building applications, 17
bundles. *See also* loadable bundles
  bundle identifiers, 12
  defined, 198
  similarity with libraries and frameworks, 196
button bar
  Debug navigator, 242, 243
  Navigator area, 20, 21
buttons, adding to UI for data model, 118–119

C

c command (console), 256
C preprocessor. *See* preprocessor
call tree (Instruments), 336, 341–342
case matching, searches and, 92, 95
Categories and Protocols behaviors (Assistant), 87
C/C++, Xcode 4 and, 3
change management, unit testing and, 273
checking out, 361–363
Clang Static Analyzer. *See* static analyzer
Class behavior (Assistant), 54
classes. *See also* subclasses
  custom, for data model UI, 124
  test classes, 278
  as units, 272
cloning, 361–363
Cocoa Autolayout Guide, 66
Cocoa frameworks
  application project windows, 14
  Cocoa Bindings controller layer, 121
  Core Data and, 112
  document-based/plain applications, 13
  Xcode 4 and, 3
Cocoa Fundamentals Guide, 49
Cocoa Touch Unit Testing Bundle option, 295–296
code. *See also* source code
  adding (system framework example), 202–203
  adding (third-party framework example), 207
  adding when creating frameworks, 210–211
  code completion, 256
  code profiling tool. *See* Instruments
  code-folding ribbon, 82
  units, defined, 272
  writing with Source Editor. *See* Source Editor, writing code with
Code Signing and Application Sandboxing Guide, 191
Code Snippet library, 85–86
colors, of issues, 100
`com.apple.main-thread` dispatch queue, 244
command sets, managing, 152
Command+Q for quitting iOS applications, 262
commands
  categories of (debugger help), 257
  console execution control commands, 256
committing
  changes to repositories (SCM), 366–367
  commits, Git vs. SVN, 356
community Web sites for help, 42
Company Identifier field, 12
company name, changing, 153
compartmentalization (nibs), 49
Compile Sources phase (targets), 170
compile time
  compile-time debugging, 100–101
  vs. runtime, 322
conditionals, preprocessor and, 315–316
`CONFIGURATION` variable, 310–311
configurations
  Build Configuration pop-up, 184
  build systems, 157
  Configurations group (targets), 160
  per-configuration settings (targets), 166
configuring snapshots, 352–353
console. *See* debugger console
constraints
  Autolayout, 64–67, 68, 69
  layout, 63
containers for schemes, changing, 227
continue-to-here, debugging and, 108
controllers
  controller objects (nibs), 48
  creating for data model UI, 121
controls
  Apps controls (build systems), 193
  Debugger Bar, 103
  hardware controls (build systems), 192
  memory view, 248, 249
  positioning, 60–61
  schemes, 182
  sizing, 63
  wiring for data model UI, 122

Convert to Objective-C 2.0
option, 148
Copy Bundle Resources phase,
170, 177
Copy Files phase (targets), 171
Copy Frameworks build phase, 224
Copy Headers build phase, 212, 213
Copy Headers phase (targets), 172
Copy PlugIns build phase, 171
copyrights, company name and, 153
Core Data, 112–113
Counterparts behavior (Assistant), 87
CPP (C preprocessor). *See*
preprocessor
"Create a branch" sheet, 369
Create Snapshot sheet, 353
Create Superclass option, refactoring
and, 147–148
custom classes, creating for data
model UI, 124
custom find scopes, 92–94
customizing
behaviors, 229–230
debug behavior, 32
shortcuts, 151–152

D

dangling references, tracking down,
345–347
Data Model Editor, 111–125
Assistant and, 28, 125
basic data model, creating, 117
basic use of, 115–116
Core Data, 112–113
UI for data model, creating. *See*
user interface for data model,
creating
date
date formatter, adding to data
model UI, 120
due date (data model UI), 117,
118, 122
Debug area, 20, 32, 102
Debug configuration, 157
Debug navigator, 24, 109, 242–243
Debugger Bar, 103–104
debugger console, 250–257
backtrace, viewing, 254–255
basics, 104–105, 250
code completion, 256
help, 257
objects and values, printing,
251–254
program execution, controlling,
256
debugging and analysis, 99–109,
233–269
for apps for iOS devices. *See* iOS
devices, debugging apps for;
iOS Simulator
Beta release scheme and, 321
compile-time debugging, 100–101
Debug navigator, 242–243
debugger console. *See* debugger
console
generated output, viewing,
258–259
logic errors (static analyzer), 241
memory and, 246–249
memory leaks (static analyzer), 236
memory over-releases (static
analyzer), 237–240
runtime debugging. *See* runtime
debugging
scripts, 305
threads and stacks, 242–245
using (static analyzer), 234–235
`#define` directive, 313
defining macros, 313–314
definitions, jumping to, 152
deleting
files, 77
folders, 363
snapshots, 354
dependencies
automatic detection, 218
defining, 185
Find Implicit Dependencies
option, 183
inter-project dependencies,
222–225
new targets, adding and, 176
subprojects and, 222
Target Dependencies build phase,
169, 177
deploying applications. *See*
applications, deploying
Deployment Target group
(targets), 160
`-description` method, 108
design of software, unit testing
and, 273
Detail view (Instruments), 335–336
developer forums (Apple), 42
developer tools documentation,
viewing, 39
Diagnostics tab (Run action), 186
discarding local changes (SCM), 373
dispatch queues, 244–245
document icons, setting, 133
document types
custom, 133
setting icons, 133
targets, 164
documentation
Documentation and API Reference
menu item, 38
Documentation mode for Help,
39–40
reporting Apple documentation
errors, 42
searching for text, 41
document-based applications, 13
downloading Xcode 4, 4
drag and drop
for adding files to projects, 72, 73
adding frameworks to projects
with, 204
projects into workspaces and,
222, 223
script files, 304

snippets, 85
dSYM files, archiving and, 136
DTrace, 5, 326
`.dylib` extension for libraries, 196
dynamic libraries, 196

E

Edit All in Scope command, 146, 149
editing
breakpoints, 24
snippets, 86
editing schemes, 181–190
Analyze action, 189
Archive action, 189
Build action, 183–184
Pre- and Post-actions scripts, 190
Profile action, 188
Run action, 184–186
scheme controls, 182
Test action, 186–187
Editor area
basics, 20, 27–29
Interface Builder, 51–52
editors
in Editor area, 27
split pane editors. *See* Assistant
embedding frameworks, 206
Encapsulate, refactoring and, 148
entities
adding, 117
Core Data and, 114
entity subclasses, creating, 124
entitlements (build systems), 191–193
environment variables, scripting and, 301, 305–306
errors, navigable, 283
events
common device events (iOS Simulator), 266–267
multi-touch events (iOS Simulator), 263
execution pointer, debugging and, 109
Explore mode (Organizer), 40
exporting
schemes, 180
snapshots, 354
expressions, testing, 280
Extended Detail view (Instruments), 336
Extract option, refactoring and, 147–148

F

File Template library, 76, 278, 287
files
Copy Files phase (targets), 171
entitlements, 192
File System pop-up (build systems), 192
files structure, built frameworks, 214
locating project files with Finder, 222
opening in Assistant, 29
owners (nibs), 48
searching within, 97
Xcode source files, 20
files, adding to projects, 71–76
creating new files, 74–75
existing files, 72–73
File Template library, 76
removing files, 77
File's Owner, 28, 48
`fin` command (console), 256
Find panel (Source Editor), 97
Find Scopes sheet, 92
Finder
finding archives and, 137
locating project files with, 222
viewing packages in, 138
finding
archives, 137
Find In option (Search navigator), 92
Find options (Search navigator), 91–92
leaks with Instruments, 342–344
projects' targets, 160
system-defined symbols and macros, 314
Fix-it feature, 101
focusing on scopes, 82
folders
folder structure, built frameworks, 214
Folders option, for adding files, 73
managing in SVN, 363
media folders (build systems), 193
folding code, 83
`Foo.framework`, 204–207
`Foundation.framework`, 197
Framework Programming Guide, 206
frameworks
basics, 197, 199
Framework and Library sheet, 199
linking targets to, 161
similarity to libraries and bundles, 196
system framework example. *See* system framework example
third-party example, 204–207
frameworks, creating, 208–215
code, adding, 210–211
headers, configuring, 212–215
new framework projects, creating, 208–209
Free Pascal, 174

G

garbage collection, static analyzer and, 240, 347
`GCC poison` pragma directive (poison), 315
GDB debugger, 250, 254, 257
generated output, viewing, 258–259

Git
  branches, 369–370
  pushing changes and, 367
  repositories, adding, 360
  repositories, creating, 358–359
  vs. SVN, 356, 360, 361
Grand Central Dispatch, 109, 244–245
graph mode (Data Model Editor), 116

H

hardware controls (build systems), 192
headers
  configuring, when creating frameworks, 212–215
  frameworks, adding code to and, 202
  selecting correct, 207
  umbrella headers, 207, 210–211
  `WebKit.h` header, 202
Hello World action, adding, 55–56
help, 37–43
  community Web sites for, 42
  debugger console, 257
  Help menu, 38
  Organizer Documentation tab, 39–40
  Source Editor, 41
  user manual, 38
  in Utility area, 41
  Xcode help, 38
Help Book, 6, 93
Help Indexer, 6
hiding schemes, 226
Hits Must option (Search navigator), 91

I

`(IBAction)performSomeAction:(id)sender;`, 48
`(IBAction)performSomeOtherAction:(id)sender;`, 48
.icns format, 6, 127
Icon Composer, 6, 129–130
icons. *See also* application icons, customizing
  creating, 129–130
  document icons, setting, 133
  icon file formats, 127
images
  of icons, 129–130
  PNG and TIFF, 128
importing schemes, 180
In-Call Status Bar (iOS devices), 267
Include Unit Tests option, 13, 208
Includes and Included By behaviors (Assistant), 87
indexes, project, 219
Info tab
  targets, 162–165
  Test action editor, 186, 187
  TestApp target, 318
`Info.plist` file, 162–163
Inspection Range controls (Instruments), 330
Installer packages, 6
installing
  Installer packages and, 6
  Xcode 4, 4
instances, entities and, 114
Instruments, 325–348
  basics, 5
  dangling references, 345–347
  Detail view, 335–336
  DTrace overview, 326
  Extended Detail view, 336
  finding leaks, 342–344
  instruments, defined, 326
  Instruments view, 334
  launching, 337–338
  other capabilities of, 347
  Profile action and, 188
  Strategy bar, 332–333
  templates, 188
  time profiling with, 339–342
  toolbar, 330–332
  user interface, 327–329
integration testing, 273, 274
Interface Builder, 50–54
  Assistant, 54
  Editor area, 51–52
  Interface Builder Object library, 53
  UI for data model, creating, 118
  Utility area, 53
  Xcode 4 and, 7
interfaces
  single/multiple interface windows, 19
  Source Editor, 80
iOS devices, debugging apps for, 260–269
  applications, quitting, 262
  debugging, 268
  iOS Developer Program, 260
  iOS project, creating, 260–262
  iOS Simulator. *See* iOS Simulator
iOS Simulator, 263–268
  basics, 5, 263
  debugger and, 268
  devices and versions, 264
  multi-touch events, 263
  rotation, 264–266
  simulating common device events, 266–267
  simulating TV Out, 267
iPads
  hooking to projectors, 267
  iOS Simulator and, 264
  iPad Simulators run destinations, 262
iPhones. *See also* iOS devices, debugging apps for
  In-Call Status Bar, 267
  project settings, 261
  run destinations, 262, 263
iPhoto app (simulated), 267
Issue navigator
  basics, 23
  failures in, 291
  navigable errors and, 283
  warnings, 15

issues
  Activity Viewer and, 33
  basics, 100
  defined, 23, 100
  logic issues, 241
  static analyzer and, 234–235

J

Jump Bar
  basics, 26
  editing source code and, 81
Jump to Definition, 152

K

key bindings, changing, 151–152
keyboard shortcuts
  Assistant panes, opening with, 29
  behaviors shortcuts, 229, 230
  breakpoints, setting with, 106
  customizing, 151–152
  tab use, speeding up with, 34

L

launching Instruments, 337–338
layout
  Autolayout. *See* Autolayout
  Autosizing control, 59
  basics, 58
  constraints, 63
  data model UI, 118–120
  layout behavior (Assistant), changing, 30
  positioning, 60–61
  sizing, 62
  springs and struts, 59–63
leaks
  finding with Instruments, 342–344
  memory leaks, 236
levels settings (targets), 167–168
`libobjc.a.dylib`, 196
libraries
  basics, 196
  Code Snippet library, 85–86
  defined, 196
  File Template library, 76
  Help, 38
  ignoring unwanted, 40
  Interface Builder Object library, 53
  Link Binary With Libraries phase (targets), 170, 222
  linking targets to, 161, 170
  using, 199
Library control (Instruments), 331
Link Binary With Libraries phase, 170, 222
linking
  against frameworks, 199, 205, 224
  targets, 170
Lion
  Autolayout and, 58, 63
  document versions API, 185–186
  low memory and, 342
  upgrading to, 64
lists, filtering, 22, 23, 24
LLDB debugger, 250
loadable bundles, 198
Localizations group (targets), 161
Lock button, current memory view and, 248
Log mode (Version Editor), 372
Log navigator
  basics, 25
  debugging and, 101
  output from scripts showing in, 302
  test results log, 281–282
logic errors, static analyzer and, 241
logs, output from scripts showing in, 302
low memory warnings
  iOS devices, 266
  leaks and, 342

M

Mac OS X 10.7 (Lion)
  Autolayout, 63–64
  document versions API, 185–186
  low memory and, 342
  upgrading to, 64
macros
  defining, 150, 313–314
  preprocessor and, 316–321
  uses for, 313–314
Manage Schemes sheet, 181
managed object contexts, 114
managed object model (MOM)
  Core Data and, 113
  in Data Model Editor, 115
  in model graph mode, 116
managing
  folders in SVN, 363
  schemes (build systems), 179–181
  snapshots, 354
  source code. *See* source code management
Manual mode (Assistant), 30
manually generated output, viewing, 258
Match Case option (Search navigator), 92
memory
  debugging and, 246–249
  examining contents, 246–249
  finding leaks and, 342
  low memory warnings, 266, 342
  memory addresses, zombies and, 345–347
  memory leaks, static analyzer and, 236
  memory over-releases, static analyzer and, 237–240
  viewing, 248–249
merging changes (SCM), 368–369
methods
  testing, 272, 274, 280
  as units, 272
model layer, managing, 112

modernization of projects, 15–16
Modernize Loop option, refactoring and, 148
MOM (managed object model)
 Core Data and, 113
 in Data Model Editor, 115
 in model graph mode, 116
mouse pointer, examining variables with, 108
Move Up/Move Down option, refactoring and, 148
multi-touch events (iOS Simulator), 263

N

names
 company name, changing, 153
 testing, 286, 292
naming. *See also* renaming
 applications, 12–13
 clones, 362
navigable errors, 283
navigating projects. *See* projects, navigating
Navigator area, 20, 21–25
 Breakpoint navigator, 24
 Debug navigator, 24
 Issue navigator, 23
 Log navigator, 25
 Project navigator, 21
 Search navigator, 23
 Symbol navigator, 22
network communication (build systems), 192
New File sheet, 74
New Project options sheet, 12
New Project template sheet, 11, 260
New Scheme sheet, 180
New Target sheet, 175
`next` command (console), 256
`.nib` extension (NextStep Interface Builder), 48
nibs, 48–49
`NSManagedObject` subclass
 creating custom, 124
 generating accessor source code, 124
`NSZombie`, 345

O

objects
 printing from console, 251–253
 testing, 274, 279, 280
 unit testing and, 272
OCUnit, 278–283
 assertions, 279–280
 navigable errors, 283
 test results log, 281–282
 test targets and classes, 278
Options bar (templates), 11
Options tab (Run action), 185–186
Organizer
 preventing opening of, 309
Organizer Documentation tab for Help, 39–40
Organizer window
 basics, 35
 snapshots and, 354
orientation, changing (iOS devices), 264–266
outdated projects, modernizing, 15–16
outlets
 nibs, 48–49
 Outlets and Referencing Outlets behaviors (Assistant), 54
over-releases, memory, 237–240
owners (nibs), 48

P

`p (NSUInteger)[[googleURL scheme] length]` command, 253
PackageMaker, 6
packages, defined, 138
panes, adding and removing (Assistant), 29
Parallelize Build option, 183, 184
Pause/Continue button (Debugger Bar), 103
`.pch` file, for macros, 313
persistent stores, 113
Person class, adding for unit testing, 284–286
plug-ins, Copy PlugIns build phase, 171
PNG (.png) images, 128
`po` command, 251, 253
poison, 315
positioning (layout), 60–61
`positiveFlag` variable, 241
post-action scripts
 custom, 300–302
 schemes, 190
post-Build-action actions, 301
`pragma mark` directive, 150, 313
pre- and post-action editor, 300–301
pre- and post-actions scripts
 custom, 300–302
 schemes, 190
preprocessor, 313–322
 compile or build time vs. runtime, 322
 conditionals, 315–316
 macros, 150, 313–314, 316–321
 poison, 315
previewing replacements, 95–96
`printenv` command, environment variables and, 305
printing objects and values, 251–254
probes, defined, 326
Product Name field, 12–13
Profile action (schemes), 188
program execution, controlling, 256
Project navigator, 21, 363, 365
project options sheet, 208, 209

projects
  adding frameworks to, 204–207
  adding to workspaces, 222
  adding unit tests to, 295–297
  inter-project dependencies, 222–225
  iOS project, creating, 260–262
  linking against frameworks, 199, 205
  subprojects, defined, 222
projects, creating
  building upon and running new projects, 17
  modernization, 15–16
  new framework project, 208–210
  templates, 11
  test project, creating, 12–14
  Welcome to Xcode window, 10
projects, navigating, 19–35
  Activity Viewer, 33
  Breakpoint navigator, 24
  Debug area, 20, 32
  Debug navigator, 24
  Editor area, 20, 27–29
  Issue navigator, 23
  Jump Bar, 26
  Log navigator, 25
  Navigator area, 21–25
  Organizer window, 35
  Project navigator, 21
  Search navigator, 23
  Symbol navigator, 22
  tabs, 34
  Utility area, 20, 31
  workspace window, 20
Protocols behavior (Assistant), 87
provisioning iOS devices, debugging and, 268
pulling changes (SCM), 368–369
pushing changes (SCM), 367

Q

Quartz Composer, 6
queues, dispatch, 244–245
quitting applications, 262

R

Record and Target controls (Instruments), 330
Refactor tool, renaming symbols and, 96
refactoring
  basics, 147–149
  unit testing and, 273
References behavior (Assistant), 125
Referencing Outlets behavior (Assistant), 54
regular expression (regex) searches, 91
release builds, triggering scripts on, 310–311
Release configuration, 157
Release mode, 135–136
removing
  actions, 301
  Assistant panes, 29
  breakpoints, 24, 106
  files from projects, 77
  schemes, 180
renaming
  entities, 117
  folders, 363
  refactoring and, 147
  renaming sheets, 148, 149
  symbols, 96, 146
  tabs, 34
  unit tests, 274
reordering schemes, 180
Replace preview sheet, 95
replacing text (Search navigator), 94–96
repositories, 357–363
  basics, 357–358
  cloning and checking out, 361–363
  creating, 358–359
  manually adding, 359–361
restoring from snapshots, 355
reverting to earlier versions (SCM), 373
revision control. *See* source code management
rotation (iOS Simulator), 264–266
rules (build systems), 157
Run action (schemes), 184–186, 187
Run Archive Script build phase, 308
Run button, 17
run destinations (build systems), 157
run logs, 25
Run Script build phases
  custom scripts, 300, 303, 306, 312
  targets, 172
Runtime Classes behavior (Assistant), 125
runtime debugging, 102–109
  basics, 102
  console, 104–105
  Debug navigator, 109
  Debugger Bar, 103–104
  Source Editor and, 108–109
  using breakpoints, 106–107
runtime vs. build time, 322

S

s command (console), 256
sandboxing, 191–193
Save As sheet, 75, 221
saving
  icons, 130
  workspaces, 221
scalars, testing, 279
Scheme Editor. *See also* editing schemes
  Scheme Editor sheet, 181, 182, 319
  tests in, 274, 291
  unit testing and, 290

schemes. *See also* editing schemes
  basics, 156
  debugger attachment and, 102
  managing when building macros, 319
  in workspaces, 219, 226–227
schemes (build systems), 178–189
  basics, 178
  finding projects' schemes, 178
  managing, 179–181
  scheme manager sheet, 179
  Scheme selector control, 178
SCM systems, 356–373
  branches, creating and switching, 369–370
  committing, 366–367
  comparing and browsing history, 370–372
  folders, managing in SVN, 363
  local changes, reverting and discarding, 373
  pulling and merging changes, 368–369
  pushing changes, 367
  repositories, managing. *See* repositories
  status, checking, 364–365
scopes
  custom find scopes, 92–94
  Edit All in Scope command, 146, 149
  Find Scopes sheet, 92
  focusing on, 82
screen savers, loadable bundles and, 198
scripting
  environment variables and, 305–306
  simple example, 307
scripts
  creating, 308–309
  debugging, 305
  extending, 312
  pre- and post-action scripts, 300–302
  Run Script build phase, 300, 303–305
  triggering on release builds only, 310–311
scrubber control (Instruments), 331
scrubbing revision timeline, 371
SDKs (software development kits), Xcode 4 installation and, 4
search and replace (project members and source), 89–97
  Search navigator. *See* Search navigator
  searching within files, 97
Search control (Instruments), 331
Search navigator, 90–95
  basics, 23, 90
  custom find scopes, 92–94
  Find Options panel, 91–92
  jumping to, 90
  replacing text, 94–96
searches
  build log search results, 302
  options in Organizer, 40
  searching documentation for text, 41
  standard Help searches, 38
selection, defined, 30
`self`, printing, 252
Sent Actions behavior (Assistant), 54
`SenTestingKit.framework`, 278
Shake event (iOS devices), 267
sharing applications, 139
Shell field (Run Script action), 301, 303
shell script file template, 304
shortcuts. *See* keyboard shortcuts
`-shouldAutorotateToInterfaceOrientation:` method, 265
Show environment variables in build log check box, 303, 305, 306
Siblings behavior (Assistant), 87
sizing
  of images, 128
  layout and, 62–63
slider control (Debug navigator), 242, 243
snapping to guides for fixing constraints, 67
snapshots feature, 352–355
snippets
  creating and editing, 86
  examining and using, 85
software development kits (SDKs), Xcode 4 installation and, 4
source code
  generating accessor source code, 124
  navigating, 81–83
source code management
  Git vs. SVN, 356
  SCM systems. *See* repositories; SCM systems
  snapshots, 352–355
Source Editor
  analyzer issue highlighted in, 100
  compiler issue highlighted in, 100
  finding symbols and macros with, 314
  Help and, 41
  interacting with debugger, 108–109
  managing breakpoints in, 106
  navigable errors and, 283
  search and replace with, 97
Source Editor, writing code with, 79–87
  Assistant and, 87
  code completion, 84
  Code Snippet library, 85–86
  interface, 80
  source code, navigating, 81–83
split pane editor. *See* Assistant
Spotlight
  Include Spotlight Importer option, 13
  loadable bundles and, 198

springs and struts, 59–63
  Autosizing control, 59
  constraints, 63
  positioning, 60–61
  sizing, 62
SSH keys, setting up, 363
stacks. *See* threads and stacks
staging feature (Git), 366
`STAssertEqualObjects()` assertion, 279
static analyzer
  Analyze action and, 189
  analyzer results bar, 235
  debugging and, 100
  logic errors and, 241
  memory leaks and, 236
  memory over-releases and, 237–240
  using, 234–235
static libraries, 196
status
  SCM, checking. *See* Subversion (SVN)
  status control (Instruments), 331
Step Into button (Debugger Bar), 103
Step Out button (Debugger Bar), 103
Step Over button (Debugger Bar), 103
`STFail()` assertion, 279
Strategy bar (Instruments), 332–333
structs, testing, 279
struts. *See also* springs and struts
  defined, 59
Style option (Search navigator), 91
styles, in Data Model Editor, 116
subclasses
  creating for entities, 124
  refactoring and, 148
  Subclasses behavior (Assistant), 87
subprojects, defined, 222
Subversion (SVN)
  folders, managing in, 363
  vs. Git, 356, 360, 361
  locally hosted repositories, creating, 359
Summary tab (targets), 161–162, 163
superclasses
  refactoring and, 147–148
  Superclasses and Subclasses behaviors (Assistant), 87
SVN. *See* Subversion (SVN)
switching branches (Git), 369–370
Symbol navigator, 22
symbols
  code completion, 84
  finding, 314
  finding documentation for, 41
  moving between, 81
  renaming, 96, 146
system framework example
  code, adding, 202–203
  importance of `WebKit.framework`, 203
  linking against frameworks, 199
  UI, adding, 200–201

T

table mode (Data Model Editor)
  basics, 115, 116
  table view control, adding to UI, 118–119
tabs
  creating new, 34
  as new Xcode feature, 34
Target Dependencies build phase, 169, 177
targets, 159–177
  adding new, 175–177
  basics, 156, 159
  Build Phases tab, 169–172
  Build Rules tab, 173–174
  Build Settings tab, 165–168
  Info tab, 162–165
  project-wide settings, 160–161
  selecting projects' targets, 160
  Summary tab, 161–162, 163
  test targets, 215, 278
templates
  basics, 9
  creating new projects and, 11
  defined (DTrace), 326
  projects, creating new, 11
  target template chooser, 159
  target template sheet, 175
terminology, Git vs. SVN, 356
`TestFoo` class, 211
testing. *See also* OCUnit; unit testing
  applications, 139
  data model UI, 123
  icons, 130
  integration testing, 273, 274
  Test action (schemes), 186–187
  test case class, 287
  TestAppTouchTests test target, 295–297
  testing bundle, 295, 296
text
  replacing with Search navigator, 94–96
  searching, 91
  searching documentation for, 41
  text fields, adding, 57
third-party framework example, 204–207
threads and stacks
  Debug navigator and, 109
  debugging and, 242–245
  minimal, full, and semi-filtered stacks, 243
  suspending threads, 245
  threads vs. breakpoints, 245
Threads and Stacks navigator, 104, 105
Threads strategy (Instruments), 332–333
TIFF (.tif) images, 128
time profiling
  with Instruments, 339–342
  Time Profiler, 308–309
to-do items (basic data model), 117–118, 121, 123

toolbar (Instruments), 330–332
tools. *See* Xcode Tools
Top Level Objects behavior (Assistant), 54
tracing. *See also* DTrace
  defined, 326
TV Out, simulating (iOS Simulator), 267

U

`UIKit.framework`, 197
umbrella headers
  basics, 207, 210–211
  defined, 210
Uniform Type Identifiers (UTIs), defining, 164
unions, testing, 279
unit testing, 271–297
  adding to projects, 295–297
  basics, 271–272
  benefits of, 273
  controversy over, 275
  Include Unit Tests option, 13, 208
  vs. integration testing, 273, 274
  limitations of, 274
  OCUnit and. *See* OCUnit
  Person class, adding, 284–286
  Test action and, 187, 276
  TestAppTests, 276–277
  tests, designing, 286
  tests, passing, 292–294
  tests, testing 291
  tests, writing, 287–290
units, defined, 272
unlocking, debugging iOS devices and, 268
updating changes (SCM), 368–369
URLs, defining (targets), 165
user interface for data model, creating, 118–125
  accessors, creating, 124
  controllers, creating, 121
  custom classes, creating, 124
  layout, 118–120
  testing, 123
  wiring controls, 122
user interface (Instruments), 327–329
user interfaces (UIs), creating, 47–69
  elements, basics of adding, 55
  Hello World action, adding, 55–56
  Interface Builder. *See* Interface Builder
  layout. *See* Autolayout; layout
  nibs, 48–49
  system framework example, 200–201
  text fields, adding, 57
user manual (Help), 38
user-defined build settings, 168
Utility area
  basics, 20, 31
  Help in, 41
  Interface Builder, 53
UTIs (Uniform Type Identifiers), defining, 164

V

validating apps, 139
values, printing from console, 251, 253–254
variables
  environment variables, scripting and, 301, 305–306
  inspecting in Source Editor, 108
Variables pane, 105
Variables view (Debug area), 246
version control. *See* source code management
Version Editor, 370–371
Version Editor mode (Source Editor), 80
versions in repository, comparing and browsing history, 370–372
View controls (Instruments), 331

W

warnings
  beta warnings, 316, 319, 321, 322
  issues and, 101
  low memory warnings, 266, 342
  of outdated project settings, 16
watches, setting, 246–247
Web sites for downloading `Foo.framework`, 204
Web sites for further information
  appendixes to book, xi
  C preprocessor, 313
  Clang Static Analyzer, iv, 100
  Cocoa Autolayout Guide, 66
  Cocoa Fundamentals Guide, 49
  Code Signing and Application Sandboxing Guide, 191
  Git, 356
  Grand Central Dispatch, 244
  Help, 42
  OCUnit, 278
  `print`, 254
  SSH keys, setting up, 363
  SVN, 356
  Xcode shortcut cheat sheet, iv
`WebKit.framework`. *See* system framework example
Welcome to Xcode window, 10
windows. *See also specific* windows
  single/multiple interface windows, 19
  workspace window, 20
workspaces, 218–231
  adding, 228–230
  basics, 218–219
  defined, 218, 228
  true, 217
  when to use, 220

workspaces, creating, 221–227
  basics, 221
  inter-project dependencies, 222–225
  projects, adding to workspaces, 222
  schemes, in workspaces, 226–227
writing
  code. *See* Source Editor, writing code with
  unit tests, 287–290

X

`.xcarchive` format, 138
Xcode Archive format, 138
Xcode 4
  advantages of, xii
  installing, 4
  obtaining, 4
  relationship to Xcode Tools, 3
Xcode integrated development environment (IDE), 5
Xcode Tools
  basics, 3
  relationship to Xcode 4, 3
  tools, listed, 5–7
`.xcodeproj` files, 217
`.xib` (XML Interface Builder), 48
xibs, 48–49

Z

zipping, packages or bundles and, 139
zombies, killing, 345–347