ROB NAPIER | MUGUNTH KUMAR

iOS 5 PROGRAMMING

PUSHING THE LIMITS

Advanced Application Development
for Apple iPhone®, iPad® and iPod® Touch

Pushing the Limits with

iOS 5 Programming

Pushing the Limits with
iOS 5 Programming

ADVANCED APPLICATION DEVELOPMENT FOR APPLE iPHONE®, iPAD®, AND iPOD® TOUCH

Rob Napier and Mugunth Kumar

WILEY
A John Wiley and Sons, Ltd, Publication

This edition first published 2012

Registered office

John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, United Kingdom

A catalogue record for this book is available from the British Library.

ISBN 978-1-119-96132-1 (paperback); ISBN 978-1-119-96158-1 (ebook); 978-1-119-96159-8 (ebook); 978-1-119-96160-4 (ebook)

Set in 9.5/12 Myriad Pro Regular by Wiley Composition Services

Printed in the United States by Bind-Rite

Dedication

To Neverwood. Thanks for your patience.
Rob

To my mother who shaped the first twenty years of my life
Mugunth

Publisher's Acknowledgements

Some of the people who helped bring this book to market include the following:

Editorial and Production
VP Consumer and Technology Publishing Director: Michelle Leete
Associate Director–Book Content Management: Martin Tribe
Associate Publisher: Chris Webb
Acquisitions Editor: Chris Katsaropolous
Assistant Editor: Ellie Scott
Development Editor: Tom Dinse
Copy Editor: Maryann Steinhart
Technical Editor: Mithilesh Kumar
Editorial Manager: Jodi Jensen
Senior Project Editor: Sara Shlaer
Editorial Assistant: Leslie Saxman

Marketing
Associate Marketing Director: Louise Breinholt
Marketing Executive: Kate Parrett

Composition Services
Compositor: Wiley Indianapolis Composition Services
Proofreaders: Laura Albert, Lindsay Amones, Melissa D. Buddendeck, Melissa Cossell
Indexer: Potomac Indexing, LLC

About the Authors

Rob Napier is a builder of tree houses, hiker, and proud father. He began developing for the Mac in 2005, and picked up iPhone development when the first SDK was released, working on products such as The Daily, PandoraBoy, and Cisco Mobile. He is a major contributor to Stack Overflow and maintains the *Cocoaphony* blog (cocoaphony.com).

Mugunth Kumar is an independent iOS developer based in Singapore. He graduated in 2009 and holds a Masters degree from Nanyang Technological University, Singapore, majoring in Information Systems. He writes about mobile development, software usability, and iOS-related tutorials on his blog (blog.mugunthkumar.com). Prior to iOS development he worked for Fortune 500 companies GE and Honeywell as a software consultant on Windows and .NET platforms. His core areas of interest include programming methodologies (Object Oriented and Functional), mobile development and usability engineering. If he were not coding, he would probably be found at some exotic place capturing scenic photos of Mother Nature.

About the Technical Editor

Mithilesh Kumar is a software engineer with a passion for user interface design, Internet protocols, and virtual worlds. He likes to prototype and build applications for iOS and Mac OS X platforms. He has extensive experience in developing UI and core components for telephony clients capable of voice, video, instant messaging, presence, and voicemail.

Mithilesh graduated with a Masters degree in Computer Science from Virginia Tech with emphasis on Human-Computer Interaction. While at graduate school, he co-authored several research papers in the area of user interfaces, computer graphics and network protocols.

Authors' Acknowledgements

Rob thanks his family for giving up many evenings that he spent in the basement writing, hacking, and otherwise failing to come upstairs. Mugunth thanks his parents and friends for their support while writing this book. Thanks to Wiley for making this book possible. It went extremely well, particularly due to Sara Shlaer's continual guiding hand. Thanks to Mithilesh Kumar who made sure what we said was true, and Tom Dinse who made sure that it was intelligible. Thanks to Chris Katsaropoulos for first reaching out and getting this project rolling. Thanks to the Apple engineers who answer questions on development forums on all those still-under-NDA issues, and the whole iOS developer community who share so much. And special thanks to Steve Jobs for building toys we could build a career around.

Contents

Introduction 1

Part I: What's New 7

Chapter 1 The Brand New Stuff 9
The History of iOS 9
What's New 10
iCloud 10
LLVM 3.0 Compiler 10
Automatic Reference Counting 10
Storyboards—Draw Your Flow 11
UIKit Customization—Appearance Proxy 11
Twitter Framework and Accounts Framework 12
Other New Features 12
Newsstand Kit *12*
Core Image for Image Processing *13*
Core Image for Feature Detection *13*
Other Minor Enhancements *13*
Summary 13
Further Reading 14
Apple Documentation 14
Other Resources 14

Chapter 2 Getting Comfortable with Xcode 4 15
Getting to Know the New User Interface 16
Tabbed Editor 17
Changes to Key Bindings 18
Project Settings Editor 19
Integrated Version Control 19
Workspaces 19

All in One Window 19
Navigating the Navigators 20
Project Navigator 20
Symbol Navigator 21
Search Navigator 21
Issue Navigator 22
Debug Navigator 22
Breakpoint Navigator 22
Log Navigator 22
Help from Your Assistant 22
Integrated Interface Builder 23
Interface Builder Panels 23
Generating Code Using Assistant Editor and Integrated Interface Builder 24
LLVM Compiler 3.0: A Tryst with the Brain 24
The Clang Front End 24
I'm a Bug! Fix Me 25
Git Your Versions Here 25
Integrated Git Version Control System 25
Versions Editor 25
Git Best Practices 26
Schemes 26
Why Schemes? 27
Think of Schemes as Implementing Your Intentions 27
Creating a Scheme 27
Sharing Your Schemes 28
Build Configurations You Can Comment 29
Creating an xcconfig File 29
Refactoring the Build Configuration File 30
Xcode 4 Organizer 30
Automatic Device Provisioning 30
Viewing Crash Logs and Console NSLog Statements 31
Viewing Applications' Sandbox Data 31
Managing Repositories 31
Accessing Your Application Archives 31
Viewing Objective-C and SDK Documentation 31
Summary 32
Further Reading 32
Apple Documentation 32
WWDC Videos 32

Blogs . . . 32
Web Resources . . . 33
Books . . . 33

Part II: Getting the Most Out of Every-Day Tools . . . 35

Chapter 3 Everyday Objective-C . . . 37
Naming Conventions . . . 37
Automatic Reference Counting . . . 39
Properties . . . 42
Property Attributes . . . 44
Property Best Practices . . . 45
Private Ivars . . . 45
Accessors . . . 45
Categories and Extensions . . . 46
+load . . . 48
Category Data using Associative References . . . 49
Category Data using the Flyweight Pattern . . . 50
Class Extensions . . . 52
Formal and Informal Protocols . . . 52
Summary . . . 54
Further Reading . . . 54
Apple Documentation . . . 54
Other Resources . . . 54

Chapter 4 Hold On Loosely: Cocoa Design Patterns . . . 55
Understanding Model-View-Controller . . . 55
Using Model Classes . . . 56
Using View Classes . . . 56
Using Controller Classes . . . 57
Understanding Delegates and Data Sources . . . 58
Working with the Command Pattern . . . 59
Using Target-Action . . . 59
Using Method Signatures and Invocations . . . 60
Using Trampolines . . . 63
Using Undo . . . 66
Working with the Observer Pattern . . . 67
Working with the Singleton Pattern . . . 70
Summary . . . 73

Further Reading. 74
Apple Documentation. 74
Other Resources. 74

Chapter 5 Getting Table Views Right 75
UITableView Class Hierarchy. 75
Understanding Table Views 76
UITableViewController. 76
UITableViewCell 76
Speed Up Your Tables 77
A Word on Performance and Interface Builder 77
To Use or Not to Use Interface Builder? 77
UITableView with Subviews in a Custom UITableViewCell. 78
UITableView with a Default UITableViewCell 82
UITableView with a Custom Drawn UITableViewCell. 84
Things to Avoid in the UITableViewCell Rendering Method. 84
Custom Non-repeating Cells. 86
Advanced Table Views. 87
Pull To Refresh. 88
Infinite Scrolling 89
Inline Editing and Keyboard 91
Animating a UITableView 92
Partially Reloading Tables. 93
Practical Implementations of Table View Animations 93
Using Gesture Recognizers in Table View Cells. 94
Table View Best Practices: Writing Clean Code with Lean Controllers 95
Data Binding Guidelines 95
Multiple UITableViewControllers Inside a Single UIViewController. 96
Storyboards. 99
Getting Started with Storyboards 99
Instantiating a Storyboard 100
Loading View Controllers within a Storyboard 100
Segues 100
Passing Data 101
Returning Data 101
Instantiating Other View Controllers 102
Performing Segues Manually. 102
Building Table Views with Storyboard. 102
Static Tables 102
Prototype Cells 102

Custom Transitions 103
Another Advantage *104*
A Disadvantage *104*
Customizing Your Views Using UIAppearance Protocol **104**
Summary **105**
Further Reading **105**
Apple Documentation 105
WWDC Videos 105
Other Resources 105

Chapter 6 Better Drawing 107
iOS's Many Drawing Systems **107**
UIKit and the View Drawing Cycle **108**
View Drawing versus View Layout **110**
Custom View Drawing **111**
Drawing with UIKit 111
Paths 112
Understanding Coordinates 114
Resizing and contentMode 118
Transforms 118
Drawing with Core Graphics 121
Mixing UIKit and Core Graphics 125
Managing Graphics Contexts 125
Optimizing UIView Drawing **128**
Avoid Drawing 128
Caching and Background Drawing 128
Custom Drawing Versus Pre-Rendering 128
Pixel Alignment and Blurry Text 129
Alpha, Opaque, Hidden 130
CGLayer **131**
Summary **132**
Further Reading **132**
Apple Documentation 132
Other Resources 134

Chapter 7 Layers Like an Onion: Core Animation 135
View Animations **135**
Managing User Interaction **137**
Drawing with Layers **138**
Setting Contents Directly 140
Implementing Display 141

Custom Drawing 141
Drawing in Your Own Context 142
Moving Things Around 143
Implicit Animations 144
Explicit Animations 145
Model and Presentation 145
A Few Words on Timings 147
Into the Third Dimension 148
Decorating Your Layers 152
Auto-animate with Actions 154
Animating Custom Properties 155
Core Animation and Threads 157
Summary 157
Further Reading 157
Apple Documentation 157
Other Resources 157

Chapter 8 Tackling Those Pesky Errors 159
Error Handling Patterns 159
Assertions 160
Exceptions 162
Catching and Reporting Crashes 163
Errors and NSError 163
Error Localization 165
Error Recovery Attempter 165
Logs 168
Logging Sensitive Information 170
Getting Your Logs 170
Summary 171
Further Reading 171
Apple Documentation 171
Other Resources 171

Part III: The Right Tool for the Job 173

Chapter 9 Controlling Multitasking 175
Best Practices for Backgrounding: With Great Power Comes Great Responsibility 175
Understanding Run Loops 177
Threading 178

Developing Operation-Centric Multitasking . . . 182
Multitasking with Grand Central Dispatch . . . 183
Creating Synchronization Points with Dispatch Barriers . . . 184
Queue Targets and Priority . . . 185
New in iOS 5 . . . 186
Queue-Specific Data . . . 186
Dispatch Data . . . 187
Summary . . . 187
Further Reading . . . 188
Apple Documentation . . . 188
WWDC Sessions . . . 188
Other Resources . . . 188

Chapter 10 REST for the Weary . . . 189
The REST Philosophy . . . 190
Choosing Your Data Exchange Format . . . 190
Parsing XML on iOS . . . 190
Parsing JSON on iOS . . . 191
NSJSONSerializer . . . 192
XML Versus JSON . . . 192
Designing the Data Exchange Format . . . 193
Model Versioning . . . 193
A Hypothetical Web Service . . . 193
Important Reminders . . . 194
RESTEngine Architecture (iHotelApp Sample Code) . . . 195
NSURLConnection versus Third-Party Frameworks . . . 195
Creating the RESTEngine . . . 196
Adding Authentication to the RESTEngine . . . 196
Adding Delegates to the RESTEngine . . . 198
Authenticating Your API Calls with Access Tokens . . . 200
Canceling Requests . . . 201
Request Responses . . . 201
Key Coding JSONs . . . 202
List Versus Detail JSON Objects . . . 205
Nested JSON Objects . . . 206
Less Is More . . . 207
Error Handling . . . 207
Localization . . . 209
Handling Additional Formats Using Category Classes . . . 210
Tips to Improve Performance on iOS . . . 210

Summary 211
Further Reading. 211
Apple Documentation. 211
Other Resources. 211

Chapter 11 Batten the Hatches with Security Services 213
Understanding the iOS Sandbox. 213
Securing Network Communications 214
How Certificates Work 215
Checking Certificate Validity 218
Determining Certificate Trust 221
Employing File Protection 222
Using Keychains 224
Sharing Data with Access Groups 225
Using Encryption. 226
Overview of AES 227
Converting Passwords to Keys with PBKDF2. 227
Applying PKCS7 Padding 229
Selecting the Mode and the Initialization Vector (IV). 229
Performing One-Shot Encryption 229
Improving CommonCrypto Performance 231
Combining Encryption and Compression 235
Summary 235
Further Reading. 236
Apple Documentation. 236
WWDC Sessions 236
Other Resources. 236

Chapter 12 Running on Multiple iPlatforms and iDevices 237
Developing for Multiple Platforms 237
Configurable Target Settings: Base SDK Versus Deployment Target. 238
Configuring the Base SDK Setting *238*
Configuring the Deployment Target Setting *238*
Considerations for Multiple SDK Support:
Frameworks, Classes, and Methods 238
Framework Availability *239*
Class Availability *239*
Method Availability *240*
Checking the Availability of Frameworks, Classes, and Methods 240
Developer Documentation. *241*
Macros in iOS Header Files *241*

Detecting Device Capabilities 242
Detecting Devices and Assuming Capabilities. 242
Detecting Hardware and Sensors 242
Detecting Camera Types *243*
Detecting Whether a Photo Library Is Empty *245*
Detecting the Presence of a Camera Flash *245*
Detecting a Gyroscope *245*
Detecting a Compass or Magnetometer *246*
Detecting a Retina Display *246*
Detecting Alert Vibration Capability *246*
Detecting Remote Control Capability *247*
Detecting Phone Call Capability *247*
In App Email and SMS 247
Checking Multitasking Awareness 248
Obtaining the UIDevice+Additions Category. 248
UIRequiredDeviceCapablities. 249
Summary 249
Further Reading. 250
Apple Documentation. 250
Other Resources 250

Chapter 13 Internationalization and Localization 251
What is Localization? 251
Localizing Strings 252
Auditing for Non-Localized Strings 253
Formatting Numbers and Dates 255
Localizing Nib Files. 258
Summary 261
Further Reading. 261
Apple Documentation. 261

Chapter 14 Selling Past the Sale with In App Purchases 263
Before You Start. 263
In App Purchase Products 263
Prohibited Items. 264
Rethinking Your Business Model 265
Setting Up Products on iTunes Connect. 266
Step 1: Create a New App ID for Your App 266
Step 2: Generate Provisioning Profiles. 267
Step 3: Create the App's Product Entry 268

Step 4: Create the In App Purchase Product Entries 269
Consumables, Non-consumables, Non-Renewing Subscriptions *270*
Auto-renewable Subscriptions *270*
Step 5: Generating the Shared Secret 271
Step 6: Creating Test User Accounts 271
In App Purchase Implementation 271
Introduction to MKStoreKit 272
Why MKStoreKit? 272
Design of MKStoreKit 273
Customizing MKStoreKit 273
Initializing MKStoreKit *274*
Configuring for Use with Server Product Model *274*
Server Setup *274*
Configuring for Use with Consumables *275*
Configuring for Use with Auto-renewable Subscriptions *275*
Making the Purchase 276
Testing Your In App Purchase 276
Troubleshooting 277
Invalid Product IDs 277
Cannot Connect to iTunes Store 277
You Have Already Purchased This Product, but It's Still Not Downloaded 277
Summary 278
Further Reading 278
Apple Documentation 278
Blogs 278
Other Resources 278

Part IV: Pushing the Limits 279

Chapter 15 Cocoa's Biggest Trick: Key-Value Coding and Observing 281
Key-Value Coding 281
Setting Values with KVC 284
Traversing Properties 284
KVC and Collections 285
KVC and Dictionaries 290
KVC and Non-Objects 290

Higher-Order Messaging with KVC 290
Collection Operators 291
Key-Value Observing 291
KVO and Collections 294
How Is KVO Implemented? 295
KVO Tradeoffs 296
Summary 297
Further Reading 297
Apple Documentation 297

Chapter 16 Think Different: Blocks and Functional Programming 299
What Is a Block? 299
Why Use Functional Programming? 300
The Human Brain Versus the Microprocessor 300
Procedural Versus Functional Paradigm 300
A 'Functional' UIAlertView 300
Declaring a Block 302
Scope of Variables 303
Stack Versus Heap 303
Implementing a Block 304
Blocks-based UIAlertViews 304
Blocks-based RESTEngine 306
Blocks and Concurrency 308
Dispatch Queues in GCD 309
NSOperationQueue Versus GCD Dispatch Queue 310
Block-based Cocoa Methods 310
UIView Animations using Blocks 311
Presenting and Dismissing View Controllers 311
TweetComposer Versus In App Email/SMS 312
Dictionary Enumeration Using NSDictionary enumerateWithBlock 312
Looking for Block-based Methods 313
Supported Platforms 313
Summary 313
Further Reading 314
Apple Documentation 314
Blogs 314
Source Code References 314

Chapter 17 Going Offline 315
Reasons for Going Offline 315
Strategies for Caching 316
Methods for Storing Your Cache 316
Implementing NSKeyedArchiver *317*
Core Data *318*
Raw SQLite *318*
NSKeyedArchiver versus Core Data *318*
Cache Versioning 319
AppCache Architecture 319
Refactoring *322*
Cache Versioning 323
Invalidating the Cache 323
Creating an In-Memory Cache 325
Designing the AppCache 325
Handling Memory Warnings 327
Handling Termination and Enter Background Notifications 328
Caching Images 328
Components of ImageCache 328
Creating the ImageCache Singleton *329*
ImageFetchOperation – NSOperation Subclass *330*
Using iCloud 330
Managing Document and Key-Value Data Storage on iCloud 331
UIDocument *331*
UIManagedDocument *331*
Key-Value Data Storage *331*
Understanding the iCloud Data Store 331
Sharing Data within Apps (or App Suites) *332*
Storing Data within Your iCloud Container *332*
A Word about iCloud Backup *332*
Summary 332
Further Reading 333
Apple Documentation 333
Books 333
Other Resources 333

Chapter 18 Fancy Text Layout 335
The Normal Stuff: Fields, Views, and Labels 335
Web Views for Rich Text 336
Displaying and Accessing HTML in a Web View 336
Responding to User Interaction 337
Drawing Web Views in Scroll and Table Views 338
Rich Editing with Web Views 338
Core Text 338
Understanding Bold, Italic, and Underline 339
Attributed Strings 339
Paragraph Styles 341
Simple Layout with CTFramesetter 342
Creating Frames for Non-Contiguous Paths 343
Typesetters, Lines, Runs, and Glyphs 345
Drawing Text Along a Curve 346
Comparison of Rich Text Options 351
Third-Party Options 351
NSAttributedString-Additions-for-HTML 351
CoreTextWrapper 352
OmniUI 352
Summary 352
Further Reading 352
Apple Documentation 352
WWDC Sessions 353
Other Resources 353

Chapter 19 Building a (Core) Foundation 355
Core Foundation Types 355
Naming and Memory Management 356
Allocators 357
Introspection 358
Strings and Data 359
Constant Strings 359
Creating Strings 359
Converting to C Strings 360
Other String Operations 362
Backing Storage for Strings 362
CFData 364

Collections 364
CFArray 364
CFDictionary 365
CFSet, CFBag 365
Other Collections 365
Callbacks 366
Toll-free Bridging 367
Summary 370
Further Reading 370
Apple Documentation 370
Other Resources 370

Chapter 20 Deep Objective-C 371
Understanding Classes and Objects 371
Working with Methods and Properties 373
How Message Passing Really Works 376
Dynamic Implementations 376
Fast Forwarding 378
Normal Forwarding 382
Forwarding Failure 382
The Flavors of objc_msgSend 383
Method Swizzling 383
ISA Swizzling 386
Method Swizzling Versus ISA Swizzling 387
Summary 387
Further Reading 388
Apple Documentation 388
Other Resources 388

Index 389

Introduction

Apple has a history of alternating its releases between user-focus and developer-focus. The good news about iOS 5 is that it's all about the developers. The addition of Automatic Reference Counting (ARC) alone is worth the upgrade for developers. In one move, Apple has eliminated the number one cause of crashes in iOS applications, while making the code easier to write and faster to run. Moving to ARC is the single best thing you can do for your application. It's the most important Objective-C feature since the autorelease pool.

But iOS 5 adds many more features for the developer. From iCloud to automatic data protection, the operating system now takes care of more of the hard problems, letting developers focus on making the best apps.

Most visible to developers is the new Xcode. Some of it is better, some of it is just different, and some of it will make you crazy. It's the new game in town, though, and everyone needs to get used to it. This book will help you figure it out.

If you're ready to take on the newest Apple release and push your application to the limits, this is the book to get you there.

Who This Book Is For

This is not an introductory book. There are many books out there that will teach you Objective-C and take you step by step through Interface Builder. This is not that book. This book assumes that you have a little experience with iOS. Maybe you're self-taught, or maybe you've taken a class. You've hopefully written at least most of an application, even if you haven't submitted it yet. If you're ready to move beyond the basics, to learn the best practices and the secrets that the authors have learned from practical experience writing real applications, then this is the book for you.

This book also is not just a list of recipes. There's plenty of sample code here, but the focus is on learning how to design, code, and maintain great iOS apps. A lot of this book is about *why* rather than just *how*. You'll learn about as much about design patterns and writing reusable code as about syntax and new frameworks.

All the examples use Xcode 4. If you're not comfortable with Xcode 4 yet, don't worry. Chapter 2 is devoted to getting you up to speed.

What This Book Covers

The iOS platforms always move forward, and so does this book. Most of the examples here require iOS 5. All examples use Automatic Reference Counting. Except in a very few places, this book will not cover backward compatibility. If you've been shipping code long enough to need backward compatibility, you probably know how to deal with it. This book is about writing the best-possible apps using the best features available.

This book focuses on the iPhone 4 and iPad 2. Most topics here are applicable to the original iPad, iPod touch, iPhone 3GS, and Apple TV. At the time of writing the iPhone 5 and iPad 3 have not been released, but everything here should apply to them as well. Chapter 12 is devoted to dealing with the differences between the platforms.

How This Book Is Structured

iOS has an extremely rich set of tools, from high-level frameworks like UIKit to very low-level tools like Core Text. Often, there are several ways to achieve a goal. As a developer, how do you pick the right tool for the job?

This book separates the everyday from the special purpose, helping you pick the right solution to each problem. You'll learn why each framework exists, how the frameworks relate to each other, and when to choose one over another. Then you'll learn how to make the most of each framework for solving its type of problem.

There are four parts to this book, moving from the most common tools to the most powerful:

Part I: What's New?

If you're familiar with iOS 4, then this section quickly introduces you to the new features of iOS 5.

- **Chapter 1: The Brand New Stuff** — iOS adds a lot of new features, and here you get a quick overview of what's available.
- **Chapter 2: Getting Comfortable with Xcode 4** — Apple recently redesigned the Xcode interface, and it can take some getting used to. This chapter shows you how to get the most out of it.

Part II: Getting the Most Out of Everyday Tools

As an iOS developer, you've encountered a wide variety of common tools, from notifications to table views to animation layers. But are you using these tools to their full potential? In this part, you learn the best practices in Cocoa development from seasoned developers.

- **Chapter 3: Everyday Objective-C**—If you're ready to move to the next level in Objective-C, this chapter introduces you to the tools experienced developers use every day to improve application design, maintainability, and reusability.
- **Chapter 4: Hold On Loosely: Cocoa Design Patterns**—Cocoa relies on a number of common and consistent design patterns. You learn what they are so you can solve problems the same way Apple does.
- **Chapter 5: Getting Table Views Right**—Table views are perhaps the most complex and commonly used UI element in iOS. They are simple and elegant in design, but confusing to developers who don't understand how they work. You learn how to use them correctly and to solve some special problems like infinite scrolling.
- **Chapter 6: Better Drawing**—Custom drawing is intimidating to many new developers, but it's a key part of building beautiful and fast user interfaces. You'll discover the available drawing options from UIKit to Core Graphics, and how to optimize them to look their best while keeping them fast.

- **Chapter 7: Layers Like an Onion: Core Animation**—iOS devices have incredible facilities for animation. With a powerful GPU and the highly optimized Core Animation, you can build engaging, exciting, and intuitive interfaces. In this chapter, you go beyond the basics and learn the secrets of animation.
- **Chapter 8: Tackling Those Pesky Errors**—You try to write perfect code, but sometimes things go wrong. How your application reacts to the unexpected is what separates decent apps from extraordinary apps. You'll learn the common patterns for error handling, how to log, and how to make your code more resilient against the unexpected.

Part III: The Right Tool for the Job

There are tools that are part of nearly every application, and there are tools that you only need from time to time. In this section, you learn about those tools and techniques that are a little more specialized.

- **Chapter 9: Controlling Multitasking**—Multitasking is an important part of many applications, and you learn how to do multiple things at once while your application is running and when your application is in the background.
- **Chapter 10: REST for the Weary**—REST-based services are a mainstay of modern applications, and you learn how to best implement them in iOS.
- **Chapter 11: Batten the Hatches with Security Services**—User security and privacy are paramount today, and you learn how to protect your application and user data from attackers with the keychain, certificates, and encryption.
- **Chapter 12: Running on Multiple iPlatforms and iDevices**—The iOS landscape gets more complex every year with iPod touch, iPhone, iPad, Apple TV, and a steady stream of new editions. It's not enough just to write once, run everywhere. You need your applications to be their *best* everywhere. You'll learn how to adapt your apps to the hardware and get the most out of every platform.
- **Chapter 13: Internationalization and Localization**—While you may want to focus on a single market today, there are small things you can do to ease the transition to a global market tomorrow. Save money and headaches later, without interrupting today's development.
- **Chapter 14: Selling Past the Sale with In App Purchases**—In App Purchases are still an untapped market for many developers. Users like the add-on content, and developers love the extra revenue. You learn the best ways to make this important feature a reality in your application.

Part IV: Pushing the Limits

This section is what this book is all about. You've learned the basics. You've learned the everyday. Now push the limits with the most advanced tools available. You learn the ins and outs of deep iOS.

- **Chapter 15: Cocoa's Biggest Trick: Key-Value Observing**—Many of Apple's most powerful frameworks rely on KVO for their performance and flexibility. You learn how to leverage the flexibility and speed of KVO, as well as the trick that makes it so transparent.
- **Chapter 16: Think Different: Blocks and Functional Programming**—Many developers are still absorbing the addition of blocks to Objective-C. They're valuable for interacting with Apple frameworks, but they also open new ways of thinking about your program. Embrace a new style, and maximize its benefits in your next project.

- **Chapter 17: Going Offline**—Network programming is hard, but even harder is providing a seamless offline experience. Learn how to best cache your data and integrate it into your network engine.
- **Chapter 18: Fancy Text Layout**—From UIKit to Core Text, iOS is full of ways to display text. There's no perfect solution for displaying rich text in iOS, so it's important to learn the trade-offs so you can choose the right solution and use it correctly.
- **Chapter 19: Building a (Core) Foundation**—When you want the most powerful frameworks available on iOS, you're going to want the Core frameworks like Core Graphics, Core Animation, and Core Text. All of these rely on Core Foundation. In this chapter you learn how to work Core Foundation data types so you can leverage everything iOS has to offer.
- **Chapter 20: Deep Objective-C**—When you're ready to pull back the curtain on how Objective-C really works, this is the chapter for you. You learn how to use the Objective-C runtime directly to dynamically modify classes and methods. You also learn how Objective-C method calls are dispatched to C function calls, and how you can take control of the system to extend your programs in incredible ways.

You can skip around in this book to focus on the topics you need most. Each chapter stands alone, except for those that require Core Foundation data objects (particularly Core Graphics, Core Animation, and Core Text). Those chapters direct you to Chapter 19, "Building a (Core) Foundation," when you need that information.

What You Need to Use This Book

All examples in this book were developed with Xcode 4.2 on Mac OS X 10.7 and iOS 5. You need an Apple developer account to access most of the tools and documentation, and you need a developer license to run applications on your iOS device. Visit `http://developer.apple.com/programs/ios` to sign up.

Most of the examples in this book will run in the iOS Simulator that comes with Xcode 4.2. You can use the iOS Simulator without an Apple developer license.

There are few differences between Xcode 4.2 on Mac OS X 10.6 and 10.7, so all examples should work under 10.6.

Finding Apple Documentation

Apple provides extensive documentation at its website and within Xcode. The URLs change frequently and are often very long. This book refers to Apple documents by title rather than by URL. To find documents in Xcode, press Cmd-Option-? or click Help → Documentation and API Reference. In the Documentation Organizer, click the Search icon, type in the name of the document, and then select the document from the search results. See Figure 1 for an example of how to search for the *Coding Guidelines for Cocoa*.

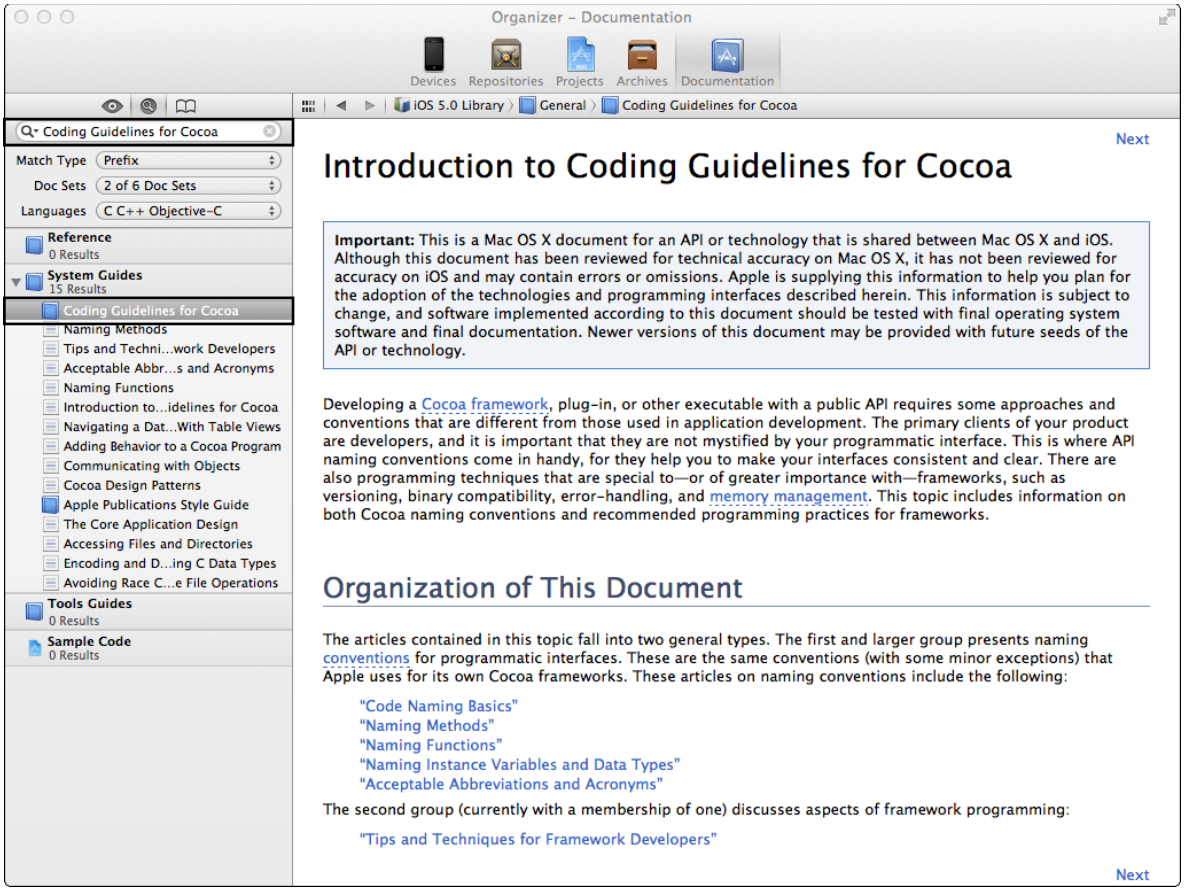

Figure 1 Searching for Coding Guidelines for Cocoa

To find documents at the Apple developer site, visit `developer.apple.com`, click Member Center and log in. Select the iOS Dev Center, and enter the document title in the Search Developer search box.

The online documentation is generally identical to the Xcode documentation. You may receive results for both iOS and Mac. Make sure to choose the iOS version. Many iOS documents are copies of their Mac counterparts, and occasionally include function calls or constants that are not available on iOS. This book guides you about which features are available on iOS.

Source Code

As you work through the examples in this book, you may choose either to type in all the code manually or to use the source code files that accompany the book. All of the source code used in this book is available for download at `www.wrox.com/go/ptl/ios5programming`. For example, you will find the following sample code online in the Chapter 18 folder, in the `SimpleLayout` project, and the `CoreTextLabel.m` file:

CoreTextLabel.m (SimpleLayout)

```
- (id)initWithFrame:(CGRect)frame {
  if ((self = [super initWithFrame:frame])) {
    CGAffineTransform
    transform = CGAffineTransformMakeScale(1, -1);
    CGAffineTransformTranslate(transform,
                               0, -self.bounds.size.height);
    self.transform = transform;
    self.backgroundColor = [UIColor whiteColor];
  }
  return self;
 }
```

Some source code snippets shown in the book are not comprehensive and are meant to help you understand the chapter. For those instances, you should refer to the files available on the website for the complete source code.