Developing and Designing Cocoa Touch Applications

Programming the

iPhone User Experience

O'REILLY®

Toby Boudreaux

Programming the iPhone User Experience

Apple's iPhone and iPod Touch not only feature the world's most powerful mobile operating system, they also usher in a new standard of human-computer interaction through gestural interfaces and multi-touch navigation. This book provides you with a hands-on, example-driven tour of UIKit, Apple's user interface toolkit, and includes common design patterns to help you create new iPhone and iPod Touch user experiences.

Using Apple's Cocoa Touch framework, you'll learn how to build applications that respond in unique ways when users tap, slide, swipe, tilt, shake, or pinch the screen. *Programming the iPhone User Experience* is a perfect companion to Apple's Human Interface Guidelines, and provides the practical information you need to develop innovative applications for the iPhone and iPod Touch.

THIS BOOK WILL HELP YOU:

- Understand the basics of the Cocoa Touch framework for building iPhone and iPod Touch applications
- Learn theory and best practices for using Cocoa Touch to develop applications with engaging and effective user interfaces
- Apply your knowledge of Objective-C to the iPhone/iPod Touch framework
- Customize standard UIKit views according to Apple's Human Interface Guidelines and usability principles
- Learn patterns for handling user experience concerns outside of the interface, such as network- and location-awareness

"Programming the iPhone User Experience *is a book the iPhone community sorely needs right now: a thoughtful inquiry into both the art and science of developing for Apple's new device. If you want to create applications that feel like they came with your iPhone, this is the book for you.*"

—Buzz Andersen
iPhone developer (Birdfeed app) and author of PodWorks, Cocoalicious, and other Macintosh software

Toby Boudreaux, CTO of The Barbarian Group, specializes in Mac and iPhone development, as well as open source web development.

INTRODUCTORY INTERMEDIATE ADVANCED

Readers should be familiar with Objective-C and Cocoa.

O'REILLY®
oreilly.com

US $34.99 **CAN $43.99**
ISBN: 978-0-596-15546-9

Programming the iPhone User Experience

Programming the iPhone User Experience

Toby Boudreaux

O'REILLY®
Beijing • Cambridge • Farnham • Köln • Sebastopol • Taipei • Tokyo

Programming the iPhone User Experience
by Toby Boudreaux

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://my.safaribooksonline.com*). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Editor: Steven Weiss
Production Editor: Sarah Schneider
Copyeditor: Emily Quill
Proofreader: Sarah Schneider

Indexer: Seth Maislin
Cover Designer: Karen Montgomery
Interior Designer: David Futato
Illustrator: Robert Romano

Printing History:
August 2009: First Edition.

ISBN: 978-0-596-15546-9

[M]

1249069920

Table of Contents

Preface

The launch of the iPhone software development kit (SDK) was a big deal for developers, designers, and consumers alike. Developers and designers were able to access a previously closed platform and distribution channel. Consumers were excited to explore an endless stream of new applications created by passionate independent developers and innovative companies.

New platforms often suffer from growing pains. Users and application creators learn simultaneously, with developers releasing applications to the market and users providing feedback. Different application teams come up with different approaches to common problems, because agreed-upon, proven solutions take time to emerge. These growing pains can be compounded when communication within the community is minimal.

In the case of the iPhone SDK, Apple has famously frustrated both developers and consumers by imposing a non-disclosure agreement (NDA) that legally restricts the ability to discuss upcoming features, tools, approaches, and technologies. To compensate for the lack of conversation within the development community, Apple provides a great set of guidelines for designing and coding iPhone applications.

The Human Interface Guidelines (HIG) describe the way applications should look and feel on Apple platforms. For the iPhone OS, Apple released a separate version of the HIG that focuses on mobile, Multi-Touch applications. The HIG works well in many regards, and it remains a valuable resource for anyone creating mobile applications.

However, the HIG cannot cover all topics that arise in the course of application development, nor can it provide insight from the market at large. Questions invariably emerge: What works for users? What causes frustration? What habits have emerged that should be avoided? What practices can help small teams or independent developers use their limited time and resources wisely? Which features should be prioritized for a shipping product? What do programmers need to know to deliver a great user experience?

You can think of this book as a supplement to the HIG—a resource that, along with Apple's extensive technical documentation, should guide teams through the choices they must make when embracing a new platform.

Audience for This Book

This book is geared toward designers, developers, and managers who wish to develop user-friendly applications for the iPhone and iPod Touch. The book mixes technical and strategic discussions, and it should be approachable by both technical developers and technology-savvy users.

The code in this book is Objective-C, and an understanding of the language is necessary to maximize the value of the code examples. If you are a desktop Cocoa developer, this book will introduce you to the differences between Cocoa and Cocoa Touch, the set of frameworks for iPhone applications. Managers and experience designers can use this book to understand the ways that applications can function together to create a holistic user experience.

Finally, this book is for readers who own and use the iPhone. To create an excellent iPhone application, a developer must have empathy for iPhone users. An appreciation of the challenges that face mobile users—both environmental and physical—is essential.

If you have no prior experience with Objective-C or Cocoa Touch, you may want to refer to an excellent book by one of this book's technical editors, Jonathan Zdziarski. His book, *iPhone SDK Application Development* (O'Reilly), provides a technical foundation in Objective-C and Cocoa Touch.

Organization of This Book

Chapter 1, *Cocoa Touch: The Core iPhone*, describes the essential information for Cocoa Touch and the devices that run the iPhone OS.

Chapter 2, *The Mobile HIG*, gives an introduction to the Human Interface Guidelines and elaborates on the most important concepts in the iPhone user experience.

Chapter 3, *Types of Cocoa Touch Applications*, presents a vocabulary for describing families of applications for the iPhone and links each to a structural application type.

Chapter 4, *Choosing an Application Template*, examines the application templates supplied with Xcode and the iPhone SDK. The concept of view controllers is explained with each type of standard view controller covered.

Chapter 5, *Cooperative Single-Tasking*, breaks from the application structure and focuses on the ways applications can work together to create a holistic user experience.

Chapter 6, *Touch Patterns*, teaches you how to work with the Multi-Touch interface, including design patterns for standard and custom gestures.

Chapter 7, *Interaction Patterns and Controls*, covers the types of user interface controls included in the Cocoa Touch UI framework, and the design patterns used to enable controls to work together.

Chapter 8, *Progressive Enhancement*, discusses techniques to layer functionality around user ability. Networking, data management, rotation, and audio functionality are addressed.

Chapter 9, *UX Anti-Patterns*, covers a set of common approaches that can cause issues for users.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic
: Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`
: Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, keywords, classes, and frameworks.

`Constant width bold`
: Used for emphasis within code examples.

This icon signifies a tip, suggestion, or general note.

Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Programming the iPhone User Experience* by Toby Boudreaux. Copyright 2009 Toby Boudreaux, 978-0-596-15546-9."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

Safari® Books Online

When you see a Safari® Books Online icon on the cover of your favorite technology book, that means the book is available online through the O'Reilly Network Safari Bookshelf.

Safari offers a solution that's better than e-books. It's a virtual library that lets you easily search thousands of top tech books, cut and paste code samples, download chapters, and find quick answers when you need the most accurate, current information. Try it for free at *http://my.safaribooksonline.com*.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at:

http://www.oreilly.com/catalog/9780596155469

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about our books, conferences, Resource Centers, and the O'Reilly Network, see our website at:

http://www.oreilly.com

Acknowledgments

I would like to thank Steve Weiss and Robyn Thomas, my editors at O'Reilly Media, for their guidance throughout this project. I would also like to thank Chandler McWilliams and Jonathan Zdziarski for their careful technical reviews, which made this a more accurate and comprehensive book. Finally, thanks to my wife and son for their flexibility and support, and for sacrificing weekend trips as I worked on the project.

CHAPTER 1

Cocoa Touch: The Core iPhone

Cocoa is a collection of tools—libraries, frameworks, and APIs—used to build applications for the Mac OS. Most of the core functionality you would need to develop a rich Mac application is included in Cocoa. There are mechanisms for drawing to display, working with text, saving and opening data files, talking to the operating system, and even talking to other computers across a network. The look and feel of Mac applications is recognizable and relatively consistent in large part because of the breadth and quality of the Cocoa user interface framework.

The Cocoa frameworks include two areas of focus: classes that represent user interface objects and collect user input, and classes that simplify challenges like memory management, networking, filesystem operations, and time management.

Developing applications for the iPhone and iPod Touch is similar in many ways to building applications for Mac OS X. The same tools are used for writing and debugging code, laying out visual interfaces, and profiling performance, but mobile application development requires a supplemental set of software libraries and tools, called the iPhone SDK (software development kit).

Cocoa Touch is a modified version of Cocoa with device-specific libraries for the iPhone and iPod Touch. Cocoa Touch works in conjunction with other layers in the iPhone and iPod Touch operating systems and is the primary focus of this book.

Mac Frameworks

Mac OS X programmers use a framework called `AppKit` that supplies all the windows, buttons, menus, graphics contexts, and event handling mechanisms that have come to define the OS X experience. The Cocoa Touch equivalent is called `UIKit`. In addition to user interface elements, `UIKit` provides event handling mechanisms and handles drawing to the screen. `UIKit` is a very rich framework and is a major focus of user experience programmers. Nearly all user interface needs are accounted for in `UIKit`, and developers can create custom UI elements very easily. Many of the user experience problems and patterns addressed in this book will focus on `UIKit` programming with an emphasis on standard solutions.

The second Cocoa Touch framework is the `Foundation` framework. You can think of `Foundation` as the layer that abstracts many of the underlying operating system elements such as primitive types, bundle management, file operations, and networking from the user interface objects in `UIKit`. In other words, `Foundation` is the gateway to everything not explicitly part of the user interface. As you'll see in this book, user experience programming goes deeper than the user interface controls and includes things such as latency management, error handling, data caching, and data persistence.

UIKit Overview

The user interface comprises the elements of a device or application that users see, click, and—in the case of Cocoa Touch—tilt, shake, or tap. User interfaces are a big part of user experience. They provide the *face* of your product, and often a bit more.

For the most part, `UIKit` is just a limited subset of the `AppKit` framework for Mac OS X. If you have experience developing Cocoa apps for the Mac, you will get your head around `UIKit` fairly quickly. The main differences are that `UIKit` is tuned for specific hardware interfaces and that it provides less functionality than `AppKit`. The reduced scope of `UIKit` is primarily due to the differences in robustness between typical computers and the iPhone or iPod Touch. Despite the omission of a few familiar elements, `UIKit` is a very capable toolset.

The best way to understand the breadth of `UIKit` is with a visual topology of the framework. Figures 1-1 and 1-2 show the layout of `UIKit`.

The core class from which all Cocoa objects inherit basic behavior is `NSObject`. The `NS` prefix has roots in the non-Apple origins of Cocoa at NeXT. The early versions of what is now Cocoa were called NextStep. Most Cocoa classes in Cocoa are subclasses of `NSObject`, and many classes assume that `NSObject` is the foundation of objects being passed around. For example, the class `NSArray`, which represents a collection of pointers, requires that any pointer it stores points to an `NSObject` subclass. In most cases, any custom class you create should inherit from `NSObject`.

The names of classes in Cocoa are often very descriptive. The following illustrations give an overview of the classes in `UIKit`. The purpose is to provide a view of the entire collection of classes so that developers of all experience levels can see the breadth of the frameworks.

In `UIKit`, all classes that respond to user input inherit from `UIResponder`, which is an `NSObject` subclass that provides functionality around handling user input. Figure 1-1 focuses on the subclasses of `UIResponder`.

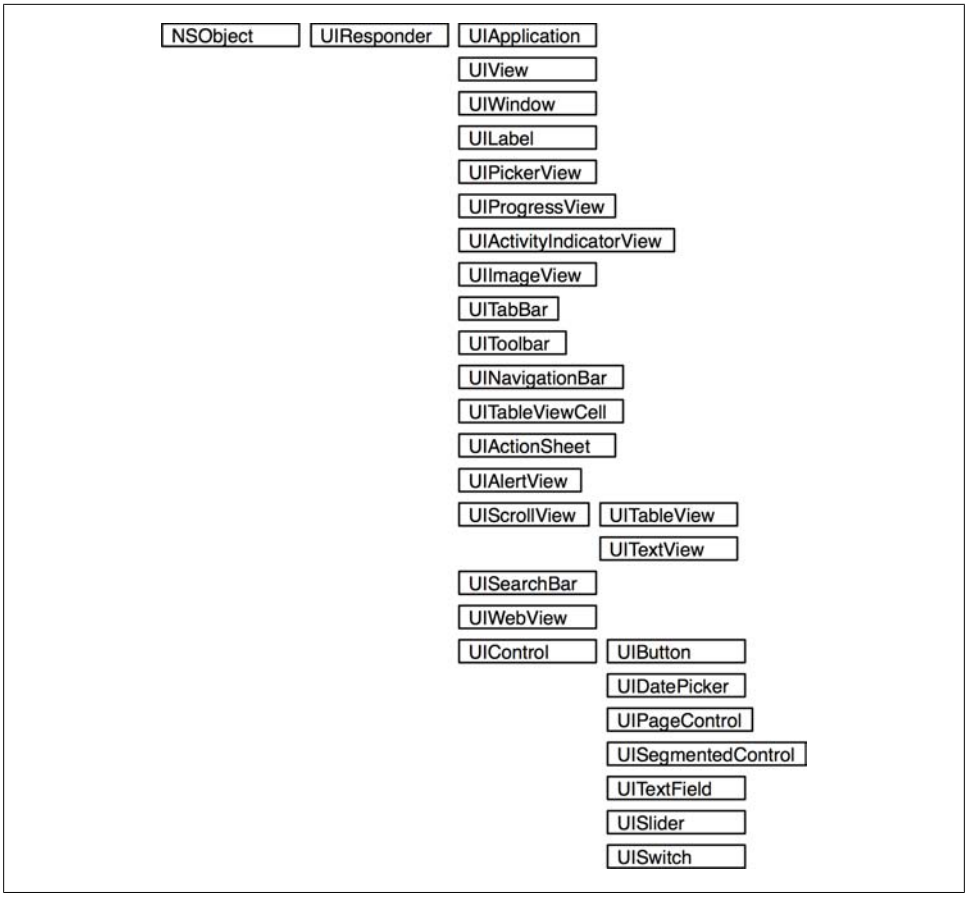

Figure 1-1. UIKit classes: UIResponder tree

In addition to the `UIResponder` class hierarchy, `UIKit` includes a set of classes acting as value objects, logical controllers, and abstractions of hardware features. Figure 1-2 shows these classes.

The documentation sets that accompany the iPhone SDK, in tandem with the Apple Developer Connection website, cover all the details of the Cocoa and Cocoa Touch framework classes. This book will further elaborate on key classes in the context of interaction design patterns and overall user experience, but the official documentation should remain the primary reference for developers, as each update to the iPhone SDK includes amended or edited documentation packages.

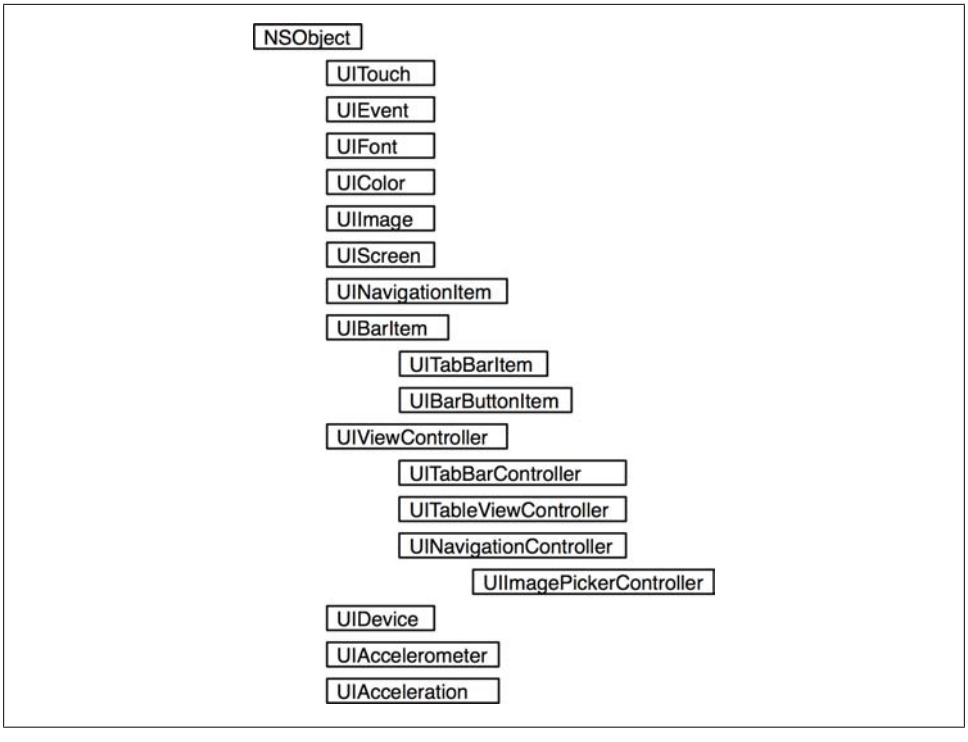

Figure 1-2. UIKit classes: controllers, value objects, device classes

Foundation Overview

The `Foundation` layer of Cocoa Touch (and Cocoa on the Mac) provides an object-oriented abstraction to the core elements of the operating system. `Foundation` handles core features of Cocoa Touch, including:

- Essential object behavior, such as memory management mechanisms
- Inter-object notification mechanisms, such as event dispatching
- Access to resource bundles (files bundled with your application)
- Internationalization and localization of resources, such as text strings and images
- Data management tools (SQLite, filesystem access)
- Object wrappers of primitive types, such as `NSInteger`, `NSFloat`, and `NSString`

All Cocoa Touch applications must link against `Foundation` because `Foundation` contains the classes that make a Cocoa application work—including many classes that are integral in the functioning of the user interface framework. For example, many `UIKit` methods use `NSString` objects as arguments or return values from methods.

The `Foundation` class tree as supported by Cocoa Touch is illustrated in Figures 1-3 to 1-9. The class hierarchy diagrams are logically grouped according to coarse functionality. This conceptual grouping mirrors the organization used by Apple in the *Cocoa Fundamentals Guide* included in the developer documentation. You should consult the *Cocoa Fundamentals Guide* and the class documentation provided by Apple as part of the iPhone SDK install for updated, in-depth information about the framework classes.

Figure 1-3 shows a subset of `NSObject` subclasses that represent value objects. Value objects are used to represent non-functional values—primitives, dates, generic binary data—and to provide utility methods for those values.

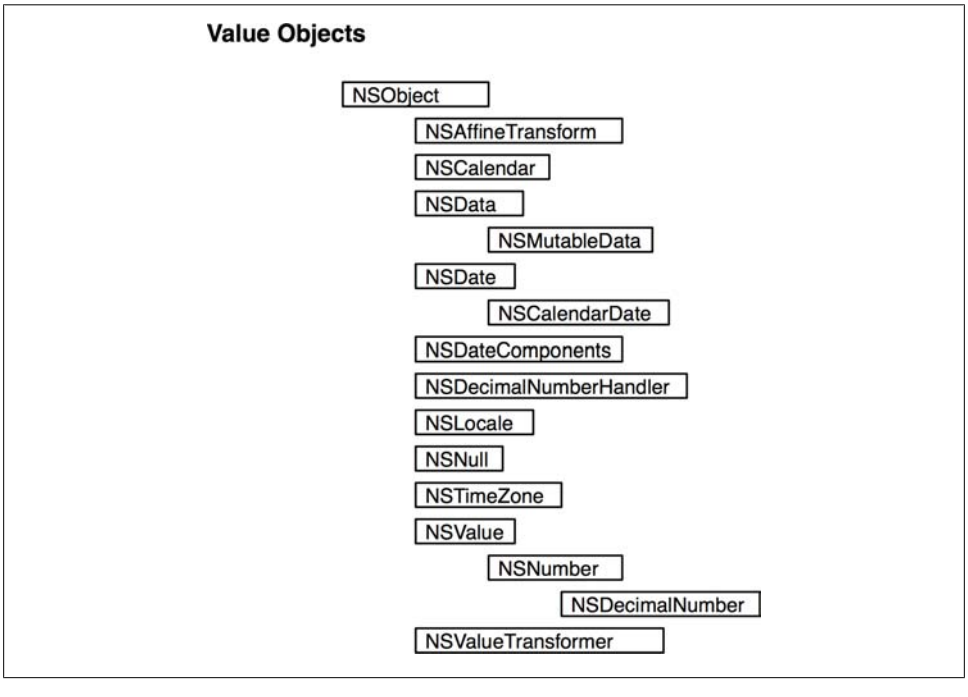

Figure 1-3. Value objects

Figure 1-4 maps the `NSObject` subclasses that focus on XML and string management. The XML classes are particularly useful when working with web services. Strings are used constantly, and Cocoa developers spend a lot of time working with `NSString` instances.

`Foundation` provides powerful classes for collection management. These classes are shown in Figure 1-5. Standard collection types such as arrays, sets, and hash tables are included, along with classes used for enumerating through collections.