Richard Wentk

Cocoa®

Developer Reference

www.wileydevreference.com

Cocoa®

Table of Contents

Cocoa®

Richard Wentk

Wiley Publishing, Inc.

Cocoa®

Published by
Wiley Publishing, Inc.
10475 Crosspoint Boulevard
Indianapolis, IN 46256
www.wiley.com

Published by Wiley Publishing, Inc., Indianapolis, Indiana

Published simultaneously in Canada

ISBN: 978-0-470-49589-6

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

Limit of Liability/Disclaimer of Warranty: The publisher and the author make no representations or warranties with respect to the accuracy or completeness of the contents of this work and specifically disclaim all warranties, including without limitation warranties of fitness for a particular purpose. No warranty may be created or extended by sales or promotional materials. The advice and strategies contained herein may not be suitable for every situation. This work is sold with the understanding that the publisher is not engaged in rendering legal, accounting, or other professional services. If professional assistance is required, the services of a competent professional person should be sought. Neither the publisher nor the author shall be liable for damages arising herefrom. The fact that an organization or Website

For general information on our other products and services or to obtain technical support, please contact our Customer Care Department within the U.S. at (877) 762-2974, outside the U.S. at (317) 572-3993 or fax (317) 572-4002.

Library of Congress Control Number: 2010935569

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

To Bea, for the inspiration.
Nam et ipsa scientia potestas est.

About the Author

With more than ten years of experience as a developer and more than fifteen years in publishing, Richard Wentk is one of Great Britain's most reliable technology writers. He covers Apple products and developments for *Macworld* and *MacFormat* magazines and also writes about technology, creativity, and business strategy for magazines such as *Computer Arts* and *Future Music*. As a trainer and a former professional Apple developer returning to development on the iPhone and OS X, he is uniquely able to clarify the key points of the development process, explain how to avoid pitfalls and bear traps, and emphasize key benefits and creative possibilities. He lives online but also has a home in Wiltshire, England. For details of apps and other book projects, visit www.zettaboom.com

Credits

Acquisitions Editor

Aaron Black

Project Editor

Martin V. Minner

Technical Editor

Benjamin Schupak

Copy Editor

Lauren Kennedy

Editorial Director

Robyn Siesky

Editorial Manager

Rosemarie Graham

Business Manager

Amy Knies

Senior Marketing Manager

Sandy Smith

Vice President and Executive Group Publisher

Richard Swadley

Vice President and Executive Publisher

Barry Pruett

Senior Project Coordinator

Kristie Rees

Graphics and Production Specialists

Joyce HaugheyJennifer Henry

Quality Control Technician

John Greenough

Proofreading

Laura Bowman

Indexing

BIM Indexing & Proofreading Services

Media Development Project Manager

Laura Moss

Media Development Assistant Project Manager

Jenny Swisher

Media Development Associate Producer

Shawn Patrick

Preface

When I started developing for the iPhone after a fifteen-year break from software, my first thought was: What is going on here? I'd written machine code for Macs and had some experience with earlier versions of Mac OS. It soon became obvious that Cocoa Touch was doing clever things behind the scenes, and that my apps were supposed to be exchanging information with those clever things.

Unfortunately, neither the official documentation nor unofficial sources of help were making it clear what those things were.

With enough persistence, it's possible for almost any developer to reverse-engineer the documentation and answer the “What is going on here?” question for himself or herself. But it's more productive to have that information before starting out. So my first goal for this book is to equip you, as a developer, with the key concepts you need to build Cocoa projects efficiently and productively.

Understanding Cocoa means more than being able to name-check concepts like delegation and Model-View-Controller; it means learning how Cocoa applies these concepts, how they influence the design of Cocoa's classes, and how your code can leverage the features built into Cocoa to simplify projects and minimize development time. In short, it means discovering how to think Cocoa. New features will begin to feel intuitive once you understand the reasoning behind them.

My second goal for the book is to give readers the skills they need to answer Cocoa questions for themselves, without handholding. OS X is vast and complex, and a full printed guide of every feature would have to be delivered on a truck. Books always sell better when readers can pick them up and take them home without stalling traffic, so this book doesn't try to detail every Cocoa feature. It also doesn't try to build complex sample projects that are unlikely to match your specific needs.

Instead, it gives you the skills you need to find answers to questions for yourself, using the official documentation and other sources of insight.

One feature you won't find in this book is cheerleading. Like any other development environment, Cocoa is a mix of excellence and unpredictability. Cocoa's best features are almost supernaturally productive and take you where you want to go with almost no code at all. Other elements offer a more scenic journey through less intuitive class relationships. Instead of a sales pitch, this book gives you a guided tour of the highlights but also warns you about some of the more dangerous parts of town.

Finally, software is as much an art as a science. Art is about creating captivating, enjoyable, and colorful experiences for an audience. In common with the Apple ethic, this book is deliberately less formal and more creative than a pure software reference. You'll find the rules here. And sometimes you'll also find suggestions for breaking the rules.

Every author tries to make his or her books as helpful as possible. Comments and feedback are welcome at `cocoadr@zettaboom.com.`

Acknowledgments

Books don't write themselves — not yet, anyway. Until operating systems become self-documenting, writing a book continues to be a team effort.

I'd like to thank acquisitions editor Aaron Black for enthusiastically suggesting the project and project editor Marty Minner for his support and for taking the manuscript and producing a book from it. Sincere thanks are also due to the rest of the team at Wiley for their hard work behind the scenes.

Software development has become a communal activity, and particular appreciation is due to the countless bloggers, experimenters, developers, and problem-solvers on the Web whose generosity and creativity have made so much possible in so many ways.

Finally — love as always for Team HGA. I couldn't have written it without you.

Introduction

This book is about developing Cocoa projects for OS X using the Xcode SDK. The theoretical elements of Cocoa are similar to those in Cocoa Touch and apply equally to both OS X and iOS. The more practical elements were written to describe OS X but with significant overlap with the equivalent features in iOS.

You'll find this book useful if you're a newcomer to Cocoa at the beginner or intermediate level and have experience with C/C++/C#, Java, Flash, Python, or a Web language such as PHP. If you're ambitious and feel up to a challenge, you can start with no experience at all. If you do, you'll find it helpful to use Objective-C (Wiley, 2010) as a companion title.

Cocoa isn't a synonym for OS X, and for practical reasons this book says little about the low-level Mach/POSIX features that underpin OS X. It mentions some of the C-level frameworks that Cocoa is built on but doesn't detail them, although it does give you enough information to explore them for yourself if you choose to.

Chapter 1 is an introduction to the history of Cocoa and OS X and explains how Cocoa evolved from Smalltalk and from the Objective-C development environment introduced by NeXT in the late 1980s. It also includes some strategic hints about the OS X and iOS application markets and how to research the current state of both so that you can target your applications for maximum return.

Chapter 2 is an informal introduction to the features that make Cocoa unique. Whether you're starting programming from scratch, or have a background in some other environment, this is one of the most critical chapters in the book. Reading it will save you time later.

Chapter 3 is a guide to the Apple documentation. It may not be obvious why this needs a guide, but Apple has structured the documentation in specific ways, and you'll progress more quickly and with less effort if you understand what this means in practice. Understanding and using the documentation is a key skill. Don't skip this chapter, even if you already have experience in other environments.

Chapter 4 explains how to join Apple's Developer Programs, and how to download and install Xcode. It also introduces the key features of Xcode 3.2.3, including the windows, menu items, and customizable toolbar. This chapter explains how to create a new sample project — an essential skill that's used repeatedly later in the book.

Chapter 5 introduces objects and classes and describes how they're implemented in Objective-C. If you have experience in other object-oriented environments, you'll need this chapter to reorient yourself to Objective-C. If you haven't, you'll find an explanation of object-oriented development that's a fundamental requirement for understanding Cocoa.

Chapter 6 explores objects in Cocoa in a more hands-on way, with very simple projects that illustrate how to use objects and their features in real Cocoa applications.

Chapter 7 introduces the key features of Interface Builder and explains how you can use IB to build complete applications, because IB isn't just for interfaces.

Chapter 8 demonstrates how to use IB to build a working application with a custom interface assembled using Cocoa library objects and how to connect a UI created in IB to code written in Xcode. This is another essential chapter. You'll need this information to build Cocoa successful applications.

Chapter 9 introduces some of the standard Cocoa design patterns and their supporting features, including target-action, Model-View-Controller, and selectors. It also looks more closely at Cocoa key-value technologies such as Key-Value Coding and Key-Value Observing and explains how to work with them effectively.

Chapter 10 introduces the Cocoa file interface and explains how it's built into many Cocoa objects, making a file manager unnecessary. For completeness, this chapter also introduces the file manager and explains how to add open and save panes to an application.

Chapter 11 explains how to manage timing, threads, and tasks in Cocoa. It also introduces the new block syntax, which is slated to replace delegation and other design patterns in future versions of OS X.

Chapter 12 introduces Cocoa's data collection classes, including `NSArray`, `NSDictionary`, and `NSSet`, and explores some of their features. It explains how to use `NSCoder` to serialize data when saving it or reloading it and introduces the essentials of both manual memory management and automated garbage collection.

Chapter 13 explores bindings, which are often seen as one of Cocoa's more challenging features but which are explained here in an unusually straightforward and practical way.

Chapter 14 follows from the previous chapter with an introduction to Core Data. It explains how to build a working card index application with no code at all and also how to customize it to make it more useful and flexible.

Chapter 15 introduces Cocoa's attributed — styled — text features and explains how to create applications with multiple document windows. You'll also find information about printing, undoing, and localizing text for foreign markets.

Chapter 16 explains how to create 2D graphics, using Cocoa's path, fill, and stroke features and also gives a low-level example of creating effects with Cocoa's Core Image filters.

Chapter 17 expands on the techniques of the previous chapter and demonstrates various animation techniques, including a simplified but animated Core Image filter. You can also find an introduction to OpenGL in Cocoa, with a sample animated teapot application.

Chapter 18 introduces various tools and strategies for debugging and profiling code and optimizing performance.

Chapter 19 is about developing for iOS. It introduces the key differences between Cocoa and Cocoa Touch, explains how to use the iOS simulator and how to get started with development on real hardware, and also explores some of the commercial opportunities offered by the iPhone and iPad.

Appendix A is about building dashboard widgets, which use JavaScript instead of Objective-C and are a quick and easy way to get started with Mac development.

Appendix B lists some of the common errors that appear in Cocoa code and introduces some possible bug-busting strategies.

Code appears `in a monospaced font`. Items you type appear in bold.

Projects were developed with Xcode 3.2.3 on OS X 10.6.3. Supporting code is available on the book's Web site at www.wiley.com/go/cocoadevref. See the readme there for the most recent system and software requirements. Code is supplied as is with no warranty and can be used in both commercial and private Cocoa projects but may not be sold or repackaged as tutorial material.

Part I: Getting Started

In This Part

Chapter 1: Introducing Cocoa

In This Chapter

Introducing Cocoa

Understanding Cocoa's history

Profiting from Cocoa

Introducing Xcode and the Apple developer programs

Apple's Cocoa technology is one of computing's success stories. When OS X 10.0 was released in 2001, it immediately revolutionized the look and feel of desktop applications. Since then, other operating systems have borrowed freely from Cocoa's innovations. Apple has continued to innovate with the iPhone and iPad, introducing Cocoa Touch for mobile devices. Cocoa Touch offers a simplified and more tactile user experience, and is the first popular and successful attempt to move beyond a traditional window, mouse, and menu interface. Future versions of Cocoa on the Mac are likely to blend the iPhone's tactile technology with the sophisticated data handling, 64-bit memory management, and rich user interface options that are already available to Cocoa developers. Cocoa is widely used in Apple's own projects, and it determines the look and feel of an application such as Aperture, shown in Figure 1.1.

Introducing Cocoa

Cocoa is the collection of libraries and design principles used to build skeleton Mac applications, create and display a user interface, and manage data. Cocoa is also a design philosophy based on unique ideas about application design and development that you can find throughout the rest of this book. You don't need to understand Cocoa's history to use the Cocoa libraries, but their features may be easier to work with if you do.

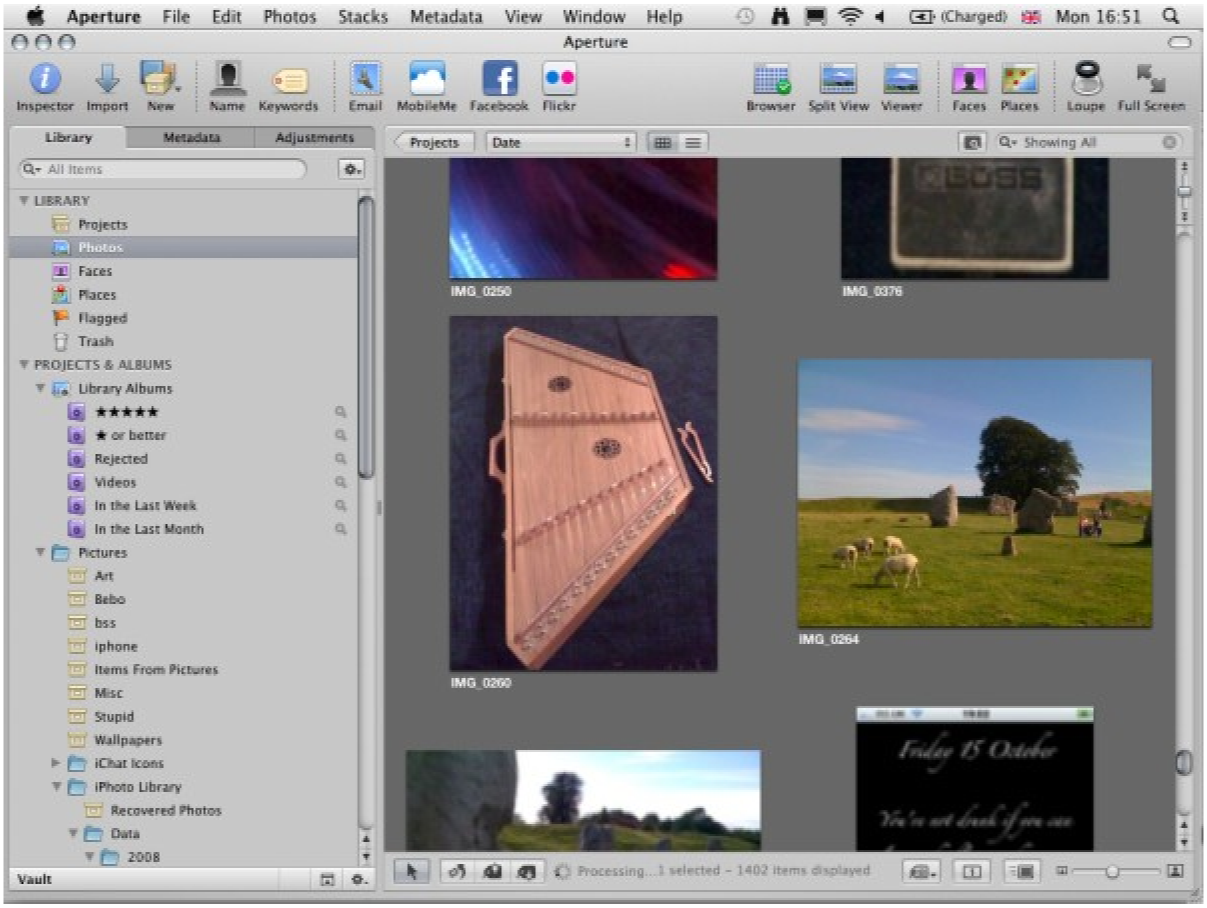
Understanding Cocoa's history

Cocoa's origins can be traced to the mid-1970s and are closely tied to the history of the Objective-C programming language. Cocoa and Objective-C are used at different levels. Cocoa is a code library and a set of interface and development guidelines. Objective-C is the language that implements them.

Cocoa is now available for other languages, including JavaScript, Python, and Ruby on Rails, but most Cocoa developers continue to work in Objective-C because its syntax and features are a natural fit for Cocoa projects.

FIGURE 1.1

Apple's Aperture application uses Cocoa technology and follows Apple's user interface design guidelines. Although Cocoa objects implement the interface, they don't enforce a standard look and feel.



Objective-C, developed by Brad Cox and Tom Love when they worked at ITT Corporation in the early 1980s, began as a mix of C and features copied from the Smalltalk experimental language. Smalltalk had been created — originally as a bet — by Alan Kay at the Xerox Palo Alto Research Center (PARC). PARC's famous graphical user interface (GUI) experiments inspired much of the visual design of both Mac OS and Windows. Smalltalk influenced those experiments by implementing a development environment in which independent objects communicated by sending and receiving messages.

At a time when most software was still *procedural* — it started at the beginning of a computer run and continued to the end, with occasional branches and subroutine calls — Smalltalk's model suggested a new and less rigid approach to software development. It enabled programmers to build applications from a library of “copy-able” but distinct interactive