Store and retrieve your app data accurately and efficiently

Pro Core Data for iOS

SECOND EDITION

Michael Privat | Robert Warner

Apress
For your convenience Apress has placed some of the front matter material after the index. Please use the Bookmarks and Contents at a Glance links to access them.
Contents at a Glance

Contents.............................................................................................................. v
About the Authors............................................................................................... x
About the Technical Reviewer ........................................................................... xi
Acknowledgments ............................................................................................ xii
Introduction ..................................................................................................... xiii
■ Chapter 1: Getting Started ............................................................................... 1
■ Chapter 2: Understanding Core Data ............................................................. 27
■ Chapter 3: Storing Data: SQLite and Other Options ....................................... 59
■ Chapter 4: Creating a Data Model ................................................................ 111
■ Chapter 5: Working with Data Objects ........................................................ 133
■ Chapter 6: Refining Result Sets ................................................................... 187
■ Chapter 7: Tuning Performance and Memory Usage ................................... 209
■ Chapter 8: Versioning and Migrating Data .................................................. 253
■ Chapter 9: Managing Table Views Using a Fetched Results Controller ....... 285
■ Chapter 10: Using Core Data in Advanced Applications .............................. 307
Index............................................................................................................... 367
Introduction

Interest in developing apps for Apple’s iOS platform continues to rise, and more great apps appear in Apple’s App Store every day. As people like you join the app-creation party, they usually discover that their apps must store data on iOS devices to be useful. Enter Pro Core Data for iOS, written for developers who have learned the basics of iOS development and are ready to dive deeper into topics surrounding data storage to take their apps from pretty good to great. Core Data, Apple’s technology for data storage and retrieval, is both easy to approach and difficult to master. This book spans the gamut, starting you with the simple and taking you through the advanced. Read each topic, understand what it means, and incorporate it into your own Core Data apps.

Why a Second Edition?

Since the publication of the first edition of Pro Core Data for iOS, Apple has released Xcode 4, a major overhaul of their programming tool. Everything has moved or changed somehow, so the descriptions and tutorials from the first edition of this book, which used Xcode 3, no longer apply. All the descriptions and screenshots have been updated to the new interface.

We didn’t stop at updating the book for Xcode 4, however. We broke the discussion of NSFetchedResultsController into its own chapter, giving it more treatment and coverage. We dug deeper into the tricky topic of migrations. We took a new approach to the section on data encryption, based on feedback from Brian Kohl. We responded to feedback we’ve received via reviews and e-mail. We think both new readers and people who have already read the first edition will profit from reading this edition.

What You’ll Need

To follow along with this book, you need an Intel Mac running Snow Leopard or Lion, and you need Xcode 4, which is available from the Mac App Store or from developer.apple.com for registered Apple developers. You’ll also do better if you have at least a basic understanding of Objective-C, Cocoa Touch, and iOS development.

What You’ll Find

This book starts by setting a clear foundation for what Core Data is and how it works, and then it takes you step-by-step through how to get the results you need from this powerful framework. You’ll learn about the components of Core Data and how they interact, how to design your data model, how to filter your results, how to tune performance, how to migrate your data across data model versions, and many other topics around and between these that will separate your apps from the crowd.
This book combines theory and code to teach its subject matter. Although you can take the book to your Barcalounger and read it from cover to cover, you'll find the book is more effective if you're in front of a computer, typing in and understanding the code it explains. We also hope that, after you read the book and work through its code, you'll keep it handy as a reference, turning to it often for answers and clarification.

How This Book Is Organized

We've tried to arrange the material so that it builds from beginning topics to advanced, at least in a general sense, as the book progresses. The topics tend to build on each other, so you'll likely benefit most by working through the book front to back, rather than skipping around. If you're looking for guidance on a specific topic—versioning and migrating data, say, or tuning performance and memory usage—skip ahead to that chapter. Most chapters focus on a single topic, indicated by that chapter's title. The final chapter covers an array of advanced topics that don't fit neatly anywhere else.

Source Code and Errata

You can and should download the source code for this book from the Apress web site at www.apress.com. Feel free to use it in your own projects, whether personal or commercial. We'll post any corrections to code as they're uncovered. We'll also post book corrections in the errata section.

How to Contact Us

We'd love to hear from you, whether it's questions, concerns, better ways of doing things, or triumphant announcements of your Core Data apps landing on the App Store. You can find us here:

Michael Privat
E-mail: mprivat@mac.com
Twitter: @michaelprivat
Blog: http://michaelprivat.com

Rob Warner
E-mail: rwarner@grailbox.com
Twitter: @hoop33
Blog: http://grailbox.com
Chapter 1

Getting Started

If you misread this book’s title, thought it discussed and deciphered core dumps, and hope it will help you debug a nasty application crash, you got the wrong book. Get a debugger, memory tools, and an appointment with the optometrist. Otherwise, you bought, borrowed, burglarized, or acquired this book somehow because you want to better understand and implement Core Data in your iOS applications. You got the right book.

You might read these words from a paper book, stout and sturdy and smelling faintly of binding glue. You might digitally flip through these pages on a nook, iPad, Kindle, Sony Reader, Kobo eReader, or some other electronic book reader. You might stare at a computer screen, whether on laptop, netbook, or monitor, reading a few words at a time while telling yourself to ignore your Twitter feed rolling CNN-like along the screen’s edge. As you read, you know that not only can you stop at any time but that you can resume at any time. Any time you want to read this book, you can pick it up. If you marked the spot where you were last reading, you can even start from where you last stopped. We take this for granted with books.

Users take it for granted with applications.

Users expect to find their data each time they launch their applications. Apple’s Core Data framework helps you ensure that they will. This chapter introduces you to Core Data, explaining what it is, how it came to be, and how to build simple Core Data-based applications for iOS. This book walks through the simplicity and complexities of Core Data. Use the information in the book to create applications that store and retrieve data reliably and efficiently so that users can depend on their data. Code carefully, though—you don’t want to write buggy code and have to deal with nasty application crashes.

What Is Core Data?

When people use computers, they expect to preserve any progress they make toward completing their tasks. Saving progress, essential to office software, code editors, and games involving small plumbers, is what programmers call persistence. Most software requires persistence, or the ability to store and retrieve data, so that users don’t have to
reenter all their data each time they use their applications. Some software can survive without any data storage or retrieval; calculators, carpenter’s levels, and apps that make annoying or obscene sounds spring to mind. Most useful applications, however, preserve some state, whether configuration-oriented data, progress toward achieving some goal, or mounds of related data that users create and care about. Understanding how to persist data to iDevices is critical to most useful iOS development.

Apple’s Core Data provides a versatile persistence framework. Core Data isn’t the only data storage option, nor is it necessarily the best option in all scenarios, but it fits well with the rest of the Cocoa Touch development framework and maps well to objects. Core Data hides most of the complexities of data storage and allows you to focus on what makes your application fun, unique, or usable.

Although Core Data can store data in a relational database (such as SQLite), it is not a database engine. It doesn’t even have to use a relational database to store its data. Though Core Data provides an entity-relationship diagramming tool, it is not a data modeler. It isn’t a data access layer like Hibernate, though it provides much of the same object-relational mapping functionality. Instead, Core Data wraps the best of all these tools into a data management framework that allows you to work with entities, attributes, and relationships in a way that resembles the object graphs you’re used to working with in normal object-oriented programming.

Early iPhone programmers didn’t have the power of the Core Data framework to store and retrieve data. The next section shows you the history behind persistence in iOS.

**History of Persistence in iOS**

Core Data evolved from a NeXT technology called Enterprise Objects Framework (EOF) by way of WebObjects, another NeXT technology that still powers parts of Apple’s web site. It debuted in 2005 as part of Mac OS X 10.4 (“Tiger”), but didn’t appear on iPhones until version 3.0 of the SDK, released in June 2009. Before Core Data, iPhone developers had the following options in terms of persistence:

- Use property lists, which contain nested lists of key/value pairs of various data types.
- Serialize objects to files using the SDK’s NSCoding protocol.
- Take advantage of the iPhone’s support for the relational database SQLite.
- Persist data to the Internet cloud.

Developers used all these mechanisms for data storage as they built the first wave of applications that flooded Apple’s App Store. Each one of these storage options remains viable, and developers continue to employ them as they build newer applications using newer SDK versions.

None of these options, however, compares favorably to the power, ease of use, and Cocoa-fitness of Core Data. Despite the invention of frameworks like FMDatabase or
ActiveRecord to make dealing with persistence on iOS easier in the pre-Core Data days, developers gratefully leapt to Core Data when it became available.

Although Core Data might not solve all persistence problems best and you might solve some of your persistence scenarios using other means like the options listed earlier, you’ll turn to Core Data more often than not. As you work through this book and learn the problems that Core Data solves and how elegantly it solves them, you’ll likely use Core Data any time you can. As new persistence opportunities arise, you won’t ask yourself, “Should I use Core Data for this?” but rather, “Is there any reason not to use Core Data?”

The next section shows you how to build a basic Core Data application using Xcode’s project templates. Even if you’ve already generated an Xcode Core Data project and know all the buttons and check boxes to click, don’t skip the next section. It explains the Core Data–related sections of code that the templates generate and forms a base of understanding on which the rest of the book builds.

**Creating a Basic Core Data Application**

The many facets, classes, and nuances of Core Data merit artful analysis and deep discussions to teach you all you need to know to gain mastery of Core Data’s complexities. Building a practical foundation to support the theory, however, is just as essential to mastery. This section builds a simple Core Data–based application using one of Xcode’s built-in templates and then dissects the most important parts of its Core Data–related code to show what they do and how they interact. At the end of this section, you will understand how this application interacts with Core Data to store and retrieve data.

**Understanding the Core Data Components**

Before building this section’s basic Core Data application, you should have a high-level understanding of the components of Core Data. Figure 1–1 illustrates the key elements of the application you will build in this section. Review this figure for a bird’s-eye view of what this application accomplishes, where all its pieces fit, and why you need them.

As a user of Core Data, you should never interact directly with the underlying persistent store. One of the fundamental principles of Core Data is that the persistent store should be abstracted from the user. A key advantage of that is the ability to seamlessly change the backing store in the future without having to modify the rest of your code. You should try to picture Core Data as a framework that manages the persistence of objects rather than thinking about databases. Not surprisingly, the objects managed by the framework must extend `NSManagedObject` and are typically referred to as, well, managed objects. Don’t think, though, that the lack of imagination in the naming conventions for the components of Core Data reveals an unimaginative or mundane framework. In fact, Core Data does an excellent job at keeping all the object graph interdependencies, optimizations, and caching in a predictable state so that you don’t have to worry about
it. If you have ever tried to build your own object management framework, you understand all the intricacies of the problem Core Data solves for you.

![Diagram of Core Data's components](image)

**Figure 1–1. Overview of Core Data’s components**

Much like we need a livable environment to subsist, managed objects must live within an environment that’s livable for them, usually referred to as a managed object context, or simply context. The context keeps track of the states of not only the object you are altering but also all the objects that depend on it or that it depends on. The `NSManagedObjectContext` object in your application provides the context and is the key property that your code must always be able to access. You typically accomplish exposing your `NSManagedObjectContext` object to your application by having your application delegate initialize it and expose it as one of its properties. Your application context will often give the `NSManagedObjectContext` object to the main view controller as well. Without the context, you will not be able to interact with Core Data.
Creating a New Project

To begin, launch Xcode, and create a new project by selecting File ➤ New ➤ New Project from the menu. Note that you can also create a new project by pressing ⌘ + ⌘ + N. From the list of application templates, select the Application item under iOS on the left, and pick Master-Detail Application on the right. Click Next, and on the next screen type BasicApplication in the Product Name field, book.coreda in the Company Identifier field, uncheck Use Storyboard and check Use Core Data. See Figure 1–2. Click the Next button, choose the parent directory where Xcode will create the BasicApplication directory and project, and click Create. Xcode creates your project, generates the project’s files, and opens its IDE window with all the files it generated, as Figure 1–3 shows.

Figure 1–2. Creating a new project with Core Data