O'REILLY®

TURING 图灵程序设计丛书

Swift与Cocoa框架开发

Swift Development with Cocoa

Jonathon Manning
[澳] Paris Buttfield-Addison 著
Tim Nugent
贾洪峰 译

中国工信出版集团
人民邮电出版社 POSTS & TELECOM PRESS

数字版权声明

图灵社区的电子书没有采用专有客户端，您可以在任意设备上，用自己喜欢的浏览器和PDF阅读器进行阅读。

但您购买的电子书仅供您个人使用，未经授权，不得进行传播。

我们愿意相信读者具有这样的良知和觉悟，与我们共同保护知识产权。

如果购买者有侵权行为，我们可能对该用户实施包括但不限于关闭该帐号等维权措施，并可能追究法律责任。

TURING 图灵程序设计丛书

Swift与Cocoa框架开发

Swift Development with Cocoa
Developing for the Mac and iOS App Stores

[澳] Jonathon Manning Paris Buttfield-Addison Tim Nugent 著
贾洪峰 译

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo

O'Reilly Media, Inc.授权人民邮电出版社出版

人民邮电出版社
北　京

图书在版编目（CIP）数据

Swift与Cocoa框架开发 /（澳）曼宁（Manning,J.），
（澳）巴特菲尔德-艾迪生（Buttfield-Addison,P.），
（澳）纽金特（Nugent,T.）著 ; 贾洪峰译. -- 北京 :
人民邮电出版社, 2015.6
（图灵程序设计丛书）
ISBN 978-7-115-39187-2

Ⅰ. ①S… Ⅱ. ①曼… ②巴… ③纽… ④贾… Ⅲ. ①
程序语言—程序设计 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2015)第095290号

内 容 提 要

本书系统讲解了在 Mac OS X 和 iOS 8 平台上，使用苹果公司的 Swift 语言开发 Mac、iPhone 和 iPad 应用的基本概念和编程技巧。主要围绕使用 Swift 语言进行 Cocoa Framework 开发，突出 OS X 和 iOS 开发的差异，教会读者利用高级 Cocoa 和 Cocoa Touch 特性开发真实的应用。

本书适合各层次 Mac OS X 和 iOS 8 应用开发人员阅读。

◆ 著 [澳] Jonathon Manning
[澳] Paris Buttfield-Addison
[澳] Tim Nugent
译 贾洪峰
责任编辑 岳新欣
责任印制 杨林杰

◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路11号
邮编 100164 电子邮件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京 印刷

◆ 开本：800×1000 1/16
印张：25.25
字数：570千字 2015年 6 月第 1 版
印数：1－3 500册 2015年 6 月北京第 1次印刷
著作权合同登记号 图字：01-2015-2572号

定价：89.00元

读者服务热线：(010)51095186转600 印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京崇工商广字第 0021 号

版权声明

O’Reilly Media, Inc.介绍

O’Reilly Media 通过图书、杂志、在线服务、调查研究和会议等方式传播创新知识。自 1978 年开始，O’Reilly 一直都是前沿发展的见证者和推动者。超级极客们正在开创着未来，而我们关注真正重要的技术趋势——通过放大那些“细微的信号”来刺激社会对新科技的应用。作为技术社区中活跃的参与者，O’Reilly 的发展充满了对创新的倡导、创造和发扬光大。

O’Reilly 为软件开发人员带来革命性的“动物书”；创建第一个商业网站（GNN）；组织了影响深远的开放源代码峰会，以至于开源软件运动以此命名；创立了 Make 杂志，从而成为 DIY 革命的主要先锋；公司一如既往地通过多种形式缔结信息与人的纽带。O’Reilly 的会议和峰会集聚了众多超级极客和高瞻远瞩的商业领袖，共同描绘出开创新产业的革命性思想。作为技术人士获取信息的选择，O’Reilly 现在还将先锋专家的知识传递给普通的计算机用户。无论是通过书籍出版，在线服务或者面授课程，每一项 O’Reilly 的产品都反映了公司不可动摇的理念——信息是激发创新的力量。

业界评论

“O’Reilly Radar 博客有口皆碑。”

——*Wired*

“O’Reilly 凭借一系列（真希望当初我也想到了）非凡想法建立了数百万美元的业务。”

——*Business 2.0*

“O’Reilly Conference 是聚集关键思想领袖的绝对典范。”

——*CRN*

“一本 O’Reilly 的书就代表一个有用、有前途、需要学习的主题。”

——*Irish Times*

“Tim 是位特立独行的商人，他不光放眼于最长远、最广阔的视野并且切实地按照 Yogi Berra 的建议去做了：‘如果你在路上遇到岔路口，走小路（岔路）。’回顾过去 Tim 似乎每一次都选择了小路，而且有几次都是一闪即逝的机会，尽管大路也不错。”

——*Linux Journal*

目录

前言

从 Mac 最早支持 Cocoa 框架开始，我们就一直在为它做开发。当时，我们见证了它由一个程序设计小环境逐渐发展成为世界上最重要、最具影响力的开发环境之一。我们之前的著作（主要介绍苹果公司的另一程序设计语言——Objective-C）每修订一次，我们都会自豪地宣称 Objective-C 在程序语言名册中的地位又得到了提升，它在 TIOBE 索引（http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html）最流行的程序设计语言中位列第三名左右。

我们还不能说 Swift（本书使用的语言）已经进入前五名（甚至还没有进入前十名），但它的名次一直在攀升，而且几乎可以肯定它将来会上升到类似的高度。

当苹果公司在 2014 年 6 月的世界开发者大会（WWDC）上发布 Swift 时，我们都为之赞叹，大感兴奋。苹果公司的主题演讲是在澳大利亚时间午夜时分进行的，等到早晨 6 点，我们就同我们那位一直都极富耐心的编辑 Rachel 召开了 Skype 会议，启动了编写本书的计划。在接下来的几个月里，我们用 Swift 开发了许多项目，目的是为了熟悉这门语言及其使用方式。

几年来，我们开发了许多复杂的大型 iOS 和 OS X 软件，交付给数千万用户使用。我们对工具集、框架和程序设计语言有着深刻的理解，而这种理解对于开发出最出色的 iOS 和 OS X 软件是至关重要的。熟悉 Swift 如何用 Cocoa 和 Cocoa Touch 完成工作很重要，所以我们花了几个月的时间来了解这一切。我们希望确保本书不只是解释 Swift，还要说明如何正确地使用 Swift 和 Cocoa/Cocoa Touch 来完成工作。

苹果公司一直在改变自己的产品，Swift 就是一例，最近推出的功能强大的 iPhone 6 和 6 Plus，以及功能日益增强的 iPad 也都能证明这一点。本书将为你提供必要的知识、信心和鉴识能力，以便用 Cocoa、Cocoa Touch 和 Swift 来进行 iOS 和 OS X 开发，还会介绍这些产品目前的工作方式。

从Objective-C到Swift

最初的 Macintosh 计算机主要用 C 语言编程，使用一个名为 Toolbox（工具箱）的库。苹果公司收购 NeXT Computer 公司以后，用 Max OS X 替换了它的操作系统，Toolbox 也被 Cocoa 代替，Cocoa 是用 Objective-C 程序设计语言编写的。

Objective-C 与其主要竞争者 C++ 的设计时间大体相同。这两门语言都是 C 程序设计语言的后代，而且都是面向对象的程序设计语言。这就是说，C 语言中的函数和数据结构是独立的，而像 C++ 和 Objective-C 这样的面向对象语言则将相关的函数和数据合并到对象中。例如，一个名为 `Car` 的对象可能包含有关其颜色、速度及开门个数的信息，可能还包含了像 `drive`、`stop` 和 `openDoor` 这样的函数。数据与处理数据的函数之间存在着紧密的联系，因而我们可以将软件看作模块化工具的集合，而不是单个大型实体。

Objective-C 是一门功能非常强大的语言。它与 C++ 的主要区别是它是一种动态语言。在任何一门面向对象的语言中，都需要将函数绑定到它们处理的具体数据，而这一绑定要么在编译代码时进行（静态绑定），要么在运行时进行（动态绑定）。C++ 使用静态绑定，这会提高运行时性能，但降低了灵活性。Objective-C 采用动态绑定，它要稍慢一些，但大大增强了语言的灵活性。

然而，Objective-C 也有自己的问题。因为它是以 C 语言为基础的，所以它从父语言那里继承了大量古怪的特性，比如预处理器和指针运算（这里只举两例）。这些功能尽管非常强大，但会降低代码的可读性和安全性。苹果公司在保持 Objective-C 的现代化方面做得非常出色，但即使是这门语言的死忠粉（本书的作者们认为自己就属于这一群体）也认为这门语言有些过时了。

于是 Swift 应运而生了。Swift 是一门新的语言，旨在让 iOS 和 Mac 应用程序的开发变得更轻松、更快速、更安全。它的设计目标是更容易掌握，且在防范程序员错误方面比 Objective-C 做得更好；事实上，苹果公司将它描述为“没有 C 的 Objective-C”。因为 Swift 是用 LLVM（Objective-C 使用的工具集）生成和编译的，而且使用了 Objective-C 运行时，所以我们可以编写一个同时使用 C、Objective-C 和 Swift 的 App。

Swift 有大量现代的程序设计语言特性，包括泛型、类型推理、类型安全、闭包、元组和自动内存管理等。Swift 是一门发展中的语言，会随着时间发生变化，但它是苹果平台软件开发的未来！在本书中，我们将学习如何在现实情景中使用 Swift，如何利用 iOS 和 OS X 中的特性让你的 App 富有吸引力。

读者对象

本书仅重点介绍 Swift，没有介绍 Objective-C 的使用。我们可能会偶尔提到后者，但并不

期望你知道如何使用它。

我们假定你是一位有能力的程序员，但并不假定你曾经为 iOS 或 OS X 做过开发，或者使用过 Swift 或 Objective-C。不过，我们假定你是一位熟练的 OS X 和 iOS 用户。

本书的组织结构

本书将讨论 Cocoa 和 Cocoa Touch，它们分别是 OS X 和 iOS 使用的框架。同时，我们还要介绍 Swift，包括其语法和功能。

几乎每一章都包含了一些可以遵照执行的实践练习。前面几章介绍了一般性主题，比如设置开发环境和 Swift 语言，后面各章将介绍 Cocoa 和 Cocoa Touch 的具体功能。

下面概述一下各章内容：

- 第 1 章　Cocoa 开发工具
  本章介绍 Cocoa 和 Cocoa Touch，也就是 OS X 和 iOS 使用的框架。还将介绍 Xcode，也就是在为这两个平台编码时使用的集成开发环境（IDE）。本章还将介绍苹果公司的开发者计划，如果想在 Mac 或 iTunes App Store 上分发软件，必须加入这一计划。

- 第 2 章　用 Swift 设计程序
  本章介绍并探索 Swift 程序设计语言、Swift 语言的特性、在用 Cocoa 和 Cocoa Touch 进行 Swift 开发时使用的设计模式。本章还将探讨基本的数据类型（比如，字符串、数组和字典）。

- 第 3 章　OS X 和 iOS 上的应用程序
  本章讨论如何生成应用程序，以及如何在 Mac 和 iOS 设备上运行它们。本章还将讨论这两种平台上的应用程序生命周期，以及沙盒如何影响应用程序对数据和资源的访问。

- 第 4 章　图形用户界面
  本章演示如何加载并向用户呈现用户界面。本章介绍了 Cocoa 提供的两个最强大的概念：nib 和故事板，它们是预先设计和预先配置的用户界面，可以直接连接到代码中。本章还谈到了 Core Animation，它是 OS X 和 iOS 使用的动画系统，还有 UI Dynamics，用于向用户界面中添加物理特性。

- 第 5 章　闭包和操作队列
  本章介绍闭包，它是 Swift 语言的一个异常强大的功能，允许将代码存储在变量中。闭包是一些函数，可以存储在变量中，也可以像值一样进行传送。这就使回调之类的功能实现起来非常轻松。本章还介绍操作队列，它是一种无需使用线程即可处理并发的简单方式。

- 第 6 章　在视图上绘制图形
  本章将学习 OS X 和 iOS 上使用的绘图系统，以及如何绘制自定义图形。我们还将介绍 Retina 显示屏，以及视图几何学是如何发挥作用的。

- 第 7 章　SpriteKit
  本章探讨 SpriteKit，它是 iOS 和 OS X 都可以使用的一种框架，用于制作快速、高效的 2D 游戏和图形。

- 第 8 章　SceneKit
  本章探讨 SceneKit，它是 iOS 和 OS X 都可以使用的一种框架，用于制作快速、高效的 3D 场景和图形。本章还将学习有关材料、约束、物理特性、相机和光照等知识。

- 第 9 章　音频与视频
  本章介绍如何使用音视频引擎 AVFoundation 进行音视频播放。还将学习如何使用语音合成、访问 iOS 照片库、访问用户照片。

- 第 10 章　iCloud 和数据存储
  本章介绍 OS X 和 iOS 上可供使用的各种存储选项。文件系统、偏好设置和 iCloud 都将进行介绍。此外，还将学习如何制作安全范围内的书签，它允许装在沙盒中的 App 持续访问用户授权 App 使用的位置。

- 第 11 章　Cocoa 绑定
  本章介绍 Cocoa 绑定，它是一种功能极为强大的系统，允许将应用程序的用户界面连接到另一应用程序的数据，而不需要中间黏合代码。绑定功能是 OS X 特有的。

- 第 12 章　表格视图和集合视图
  本章介绍表格视图（一种向用户显示多行数据的有效方式）和集合视图（可以向用户显示一组项目）。

- 第 13 章　基于文档的应用程序
  本章讨论 iOS 和 OS X 上的文档系统，它们在创建能够处理多个文档的应用程序时很有用。这里将讨论这两个平台在文档处理方式上的区别。

- 第 14 章　联网
  Cocoa 和 Cocoa Touch 为访问联网资源提供了简单易用的工具，本章演示如何从互联网提取信息，同时保持应用程序能够及时作出响应。本章还将介绍网络服务发现系统、Bonjour 和多点连接。

- 第 15 章　与现实世界互动
  本章介绍各种用于应对现实世界的技术：Core Location，用于访问 GPS；Core Motion，用于了解硬件如何移动及其方向；iOS 和 OS X 上的打印系统。还将讨论 Beacon、游戏控制器和地图。

- 第 16 章　EventKit
  本章讨论 iOS 和 OS X 上使用的日历系统，并演示如何访问用户的日历。我们还将讨论关于用户隐私的考虑事项。

- 第 17 章　Instruments 和调试器
  本章将讨论 Instruments，它是用于 Mac 和 iOS 应用程序的探查器和分析工具。本章将给出一个崩溃应用程序的例子，还将使用此应用程序诊断和纠正崩溃的原因。另外，本章将介绍 Xcode 的内置调试器。

- 第 18 章　共享与通知
  本章讨论应用程序如何利用内置的共享系统与各种服务（比如 Twitter 和 Facebook）共享文本、图像和其他内容（这些共享系统不需要你的应用程序来处理这些服务的验证）。另外，还将介绍推送通知和本地通知，利用这些方式，你的应用程序可以在未运行的情况下向用户显示信息。

- 第 19 章　非标准 App
  并非我们编写的每个程序都是驻留在用户的主（Home）屏幕上的，本章将告诉你如何编写四种不同类型的非传统 App：命令行工具、菜单栏 App、多屏 iOS App 和偏好设置窗格。

- 第 20 章　处理文本
  本章介绍 TextKit，还有 iOS 和 OS X 上可供使用的字符串本地化系统。这里还将讨论如何使用内置数据检测器从文本中提取数据。

排版约定

本书使用了下列排版约定。

- 楷体
  表示新术语或突出强调的内容。

- 等宽字体（`Constant width`）
  表示程序片段，以及正文中出现的变量、函数名、数据库、数据类型、环境变量、语句和关键字等。

- 加粗等宽字体（**`Constant width bold`**）
  表示应该由用户输入的命令或其他文本。

该图标表示提示或建议。

该图标表示一般注记。

该图标表示警告或警示。

使用代码示例

补充材料（代码示例、练习、勘误表等）可以从 http://www.secretlab.com.au/books/swift-development-with-cocoa 下载。

本书是要帮你完成工作的。一般来说，如果本书提供了示例代码，你可以把它用在你的程序或文档中。除非你使用了很大一部分代码，否则无需联系我们获得许可。比如，用本书的几个代码片段写一个程序就无需获得许可，销售或分发 O'Reilly 图书的示例光盘则需要获得许可；引用本书中的示例代码回答问题无需获得许可，将书中大量的代码放到你的产品文档中则需要获得许可。

我们很希望但并不强制要求你在引用本书内容时加上引用说明。引用说明一般包括书名、作者、出版社和 ISBN。比如："*Swift Development with Cocoa* by Jonathon Manning, Paris Buttfield-Addison, and Tim Nugent (O'Reilly). Copyright 2015 Secret Lab, 978-1-491-90894-5"。

如果你觉得自己对示例代码的用法超出了上述许可的范围，欢迎你通过 permissions@oreilly.com 与我们联系。

Safari® Books Online

Safari Books Online（http://www.safaribooksonline.com）是应运而生的数字图书馆。它同时以图书和视频的形式出版世界顶级技术和商务作家的专业作品。技术专家、软件开发人员、Web 设计师、商务人士和创意专家等，在开展调研、解决问题、学

习和认证培训时，都将 Safari Books Online 视作获取资料的首选渠道。

对于组织团体、政府机构和个人，Safari Books Online 提供各种产品组合和灵活的定价策略。用户可通过一个功能完备的数据库检索系统访问 O'Reilly Media、Prentice Hall Professional、Addison-Wesley Professional、Microsoft Press、Sams、Que、Peachpit Press、Focal Press、Cisco Press、John Wiley & Sons、Syngress、Morgan Kaufmann、IBM Redbooks、Packt、Adobe Press、FT Press、Apress、Manning、New Riders、McGraw-Hill、Jones & Bartlett、Course Technology 以及其他几十家出版社的上千种图书、培训视频和正式出版之前的书稿。要了解 Safari Books Online 的更多信息，我们网上见。

联系我们

请把对本书的意见和疑问发送给出版社。

美国：

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

中国：

北京市西城区西直门南大街 2 号成铭大厦 C 座 807 室（100035）
奥莱利技术咨询（北京）有限公司

O'Reilly 的每一本书都有专属网页，你可以在那儿找到本书的相关信息，包括勘误表、示例代码以及其他信息。本书的网站地址是：

http://shop.oreilly.com/product/0636920034285.do

对于本书的评论和技术性问题，请发送电子邮件到：bookquestions@oreilly.com

要了解更多 O'Reilly 图书、培训课程、会议和新闻的信息，请访问以下网站：

http://www.oreilly.com

我们在 Facebook 的地址如下：http://facebook.com/oreilly

请关注我们的 Twitter 动态：http://twitter.com/oreillymedia

我们的 YouTube 视频地址如下：http://www.youtube.com/oreillymedia

致谢

Jon 感谢他的父亲、母亲和整个大家庭给予自己的巨大支持。

Paris 感谢他的母亲，没有她，他就不可能完成什么有意义的事情，更不要说写书了。

Tim 感谢他的父母和整个家庭，感谢他们容忍自己缺乏活力的生活方式。

我们一起感谢我们的编辑 Brian Jepson 和 Rachel Roumeliotis，他们的才能和建议对于完成本书至关重要。同样，在编写本书的过程中与我们打过交道的所有 O’Reilly Media 工作人员都绝对是相关领域的权威。

非常感谢 Tony Gray 和 Apple University Consortium（AUC，http://www.auc.edu.au/），感谢他们给予我们及这里提到的所有人以巨大的推动力。如果不是他们，我们可能不会编写本书。

还要感谢 Neal Goldstein，是他将我们带进了这个写书的行当，所有荣誉以及（或者）过错都应归他一人。

我们还要感谢 MacLab 那帮怪人的支持（他们知道自己是什么人，并且一直守望着 Admiral Dolphin 理所当然地成为典范）。还要感谢 Christopher Lueg 教授、Leonie Ellis 博士和塔斯马尼亚大学其他容忍我们的员工。

还要因为种种原因而感谢 Nic W.、Andrew B.、Jess L. 和 Ash J.。最后，非常感谢史蒂夫·乔布斯，没有他，本书（及其他许多类似书籍）都失去了存在的理由。

第1章

Cocoa开发工具

用 Cocoa 和 Cocoa Touch 开发应用程序时，会用到由苹果公司开发的一组工具。本章将会介绍这些工具，包括如何获取它们、如何使用它们、它们如何协同工作以及它们能够做些什么。

这些开发工具的历史都很悠久，并且带有传奇色彩。它们最初是为 NeXTSTEP OS 开发的一组独立应用程序工具，最终被苹果公司用作官方的 OS X 工具。后来，苹果公司将它们中的大部分合并到一个应用程序中，称为 Xcode，但是其中一些应用程序（比如 Instruments 和 iOS Simulator）仍然保持相对独立，这主要是因为它们在开发过程中扮演着次要角色。

除了这些用于开发的应用程序之外，苹果公司还提供了开发者计划（Developer Programs，原来叫作 Apple Developer Connection，https://developer.apple.com/programs/）的成员资格。这些计划为开发人员提供了资源和支持，让那些希望与框架工程师交流的开发人员可以访问在线开发人员论坛，并获取专业技术支持。

苹果公司为 OS X 和 iOS 组织和管理着各种应用程序商店，开发人员在向 Mac App Store 或 iTunes App Store 提交应用程序时需要提供其证书，而这些开发者计划已经成为提供这些证书的官方途径。其实，它们就是通过苹果销售 App 的入场券。我们将在本章了解如何进行注册以加入这些计划，还会学习如何使用 Xcode 这一用于为 OS X 和 iOS 开发 App 的工具。

1.1 Mac和iOS开发者计划

苹果运行着两个开发者计划，分别对应着两个 App 开发平台：iOS 和 OS X。

如果希望在自己的 iOS 设备上运行代码，那就需要拥有 iOS 开发者计划（iOS Developer Program，https://developer.apple.com/programs/ios/）的付费成员资格，因为要获得必需的代码签名证书，注册是唯一途径。（在编写本书时，要获得该计划的成员资格，所需支付的费用是 99 美元 / 年。）如果并不打算向 Mac App Store 提交 App（比如，你可能更愿意自行销售 App），那就不需要成为 Mac 开发者计划（Mac Developer Program）的成员。但是，Mac 开发者计划中有一些有用的东西，比如可以提前获取 OS 的下一个版本，所以，如果你真的准备开发一些 App，那加入它还是值得的。即使你不是上述任一开发者计划的成员，也可以免费下载 Xcode。

这两个计划都提供了以下功能（当然还有其他一些小功能）。

- 访问苹果开发者论坛（Apple Developer Forums，https://developer.apple.com/devforums/）。苹果公司的工程师经常访问这些论坛，设计它们的目的就是让你向其他开发人员和编写 OS 的人提问。
- 在向公众发布 OS 的测试版本之前获得这些版本。这让你可以在 OS X 和 iOS 的下一个版本上测试自己的应用程序，提前进行必要的修改。你还可以获得开发工具的测试版本。
- 一个数字签名证书（OS X 和 iOS 各用一个），用于向 App Store 证明你的身份。没有这一证书，你就不能向 App Store 提交 App。因此，只要你希望通过 App Store 发布软件，无论是免费软件还是付费软件，都必须加入这些计划。

开发人员可以选择注册这两个开发者计划中的一个，也可以两个都注册。它们之间不存在依赖关系。

最后要说的是，查看文档或下载开发工具的当前版本并不需要注册加入某一开发者计划，因此编写 App 本身是不需要掏腰包的。

1.1.1 注册开发者计划

要注册开发者计划，首先得有一个 Apple ID。你很可能已经有一个了，因为苹果公司的大多数线上服务都需要一个 Apple ID 来验证身份。如果你曾经用过 iCloud、iTunes 商店（用于获取音乐或 App）、MobileMe 或苹果公司的支持与维修服务，那你肯定已经有了 Apple ID。你甚至可能有不止一个（本书的一位作者就有四个）。如果你还没有 Apple ID，要在注册过程中创建一个。在注册开发者计划时，它会添加到你的 Apple ID 中。

首先访问苹果公司的网站，找到要注册的计划。

- 要注册 Mac 开发者计划，请访问 http://developer.apple.com/programs/mac/。
- 要注册 iOS 开发者计划，请访问 http://developer.apple.com/programs/ios/。

接下来单击鼠标，完成各个注册步骤。

你可以选择以个人身份或公司身份进行注册。如果以个人身份注册，将你的名义销售你的App。如果以公司身份注册，将以公司依法登记的名称销售你的App。请小心选择，因为要说服苹果公司为你修改计划类型，是一件非常困难的事情。

如果以个人身份注册，只需要信用卡就行。如果以公司身份注册，除了信用卡之外，还需要相关文档来证明你有权限约束你的公司，使其遵守苹果公司的各项条款和条件。

如果想了解代码签名以及如何使用Xcode在自己的物理设备上测试和运行App，请参阅苹果公司的App Distribution Guide（https://developer.apple.com/ library/ios/documentation/IDEs/Conceptual/AppDistributionGuide/LaunchingYourApponDevices/LaunchingYourApponDevices.html#//apple_ref/doc/uid/TP40012582-CH27-SW1）。

苹果公司通常需要大约24小时来激活个人账号，公司账号需要的时间更长一些。当苹果公司向你确认后，会通过电子邮件向你发送一个用于激活账号的链接。完成激活后，你就成为一个具有完全资格的开发者了！

1.1.2 下载Xcode

要为上述两个平台中的任何一个开发App，都会使用苹果公司的集成开发环境Xcode。Xcode将源代码编辑器、调试器、编译器、探查器、iPhone与iPad模拟器等都合并到一个软件包中，在开发应用程序时，你的绝大多数时间都会花费在这里。

Xcode只能在Mac上使用。

可以从Mac App Store中获取Xcode。只需打开App Store应用程序，搜索“Xcode”，它就会弹出来。Xcode尽管非常大（编写本书时它有几个GB），但它的下载却是免费的。

下载Xcode之后，安装非常简单。Mac App Store提供了一个安装程序，只要双击该程序，然后按提示进行安装即可。

只有在使用Xcode 6或更高的版本时才能使用Swift。请确保你正在使用Xcode的最新版本。

1.2 用Xcode创建自己的第一个项目

Xcode 是围绕单窗口设计的。你的每个项目都会有一个窗口，它的大小会相应调整，以显示你正在处理的内容。

为了开始探索 Xcode，首先需要创建一个项目，步骤如下。

(1) 启动 Xcode。打开 Spotlight（按下 Command– 空格键），并键入“Xcode”，就能找到它。也可以通过以下方式找到它：打开 Finder，进入硬盘，打开“应用程序”目录。如果之前曾经打开过任何项目，Xcode 会为你打开它们。否则，将会显示 Welcome to Xcode 屏幕（见图 1-1）。

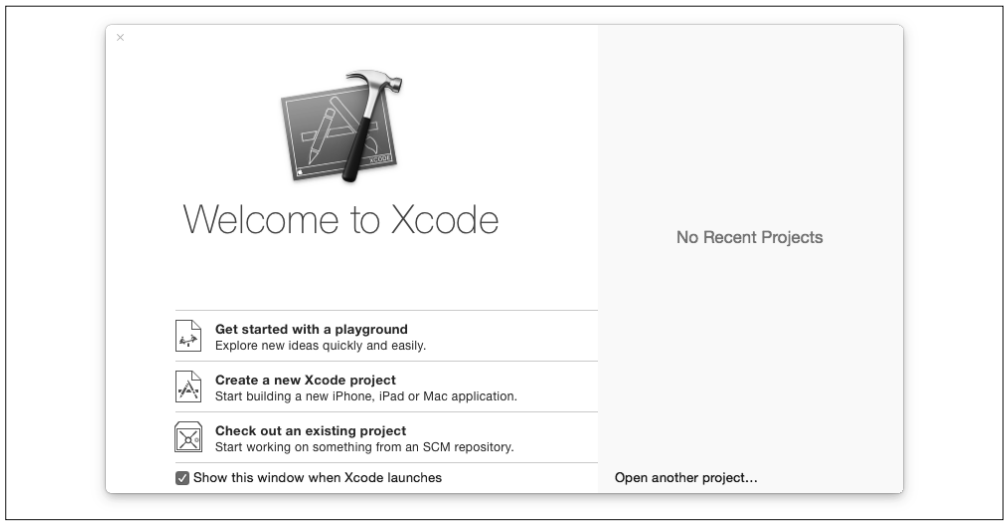

图 1-1：Welcome to Xcode 屏幕

(2) 创建一个新项目。为此，只需单击“Create a new Xcode project”（创建一个新 Xcode 项目），或者进入 File → New → Project。

Xcode 会询问你要创建哪种应用程序。模板选择器分为两个区域，左侧是一些应用程序类别。你可以选择创建一个 iOS 或 Mac 项目模板，通过这个模板能够创建一个项目目录，保证你开始工作时的方向是正确的。

因为我们现在只是随便体验一下 Xcode，具体选择哪类模板无关紧要，所以让我们选择 iOS 标题下的 Application（应用程序），然后选择 Single View Application（单视图应用程序）。这样会创建一个空的 iOS 应用程序。

(3) 输入项目的相关信息。根据选择的项目模板类型，会要求你提供一些不同的信息，用于说明应当如何配置这个新项目。

无论选择哪个平台、哪个模板，都会要求你提供如下信息。

- Product Name（*产品名称*）
  这是项目的名称，对用户可见，将来可以修改。

- Organization Identifier（*机构标识符*）
  这一信息用于生成捆绑包 ID（bundle ID），这个 ID 是一个字符串，看起来像一个前后颠倒的域名（比如，如果 O'Reilly 开发了一个名为 MyUsefulApplication 的应用程序，那捆绑包 ID 就会是 com.oreilly.MyUsefulApplication）。

捆绑包 ID 是一个应用程序的独有标识符，用于向系统和 App Store 验证该 App。因为每个捆绑包 ID 都是独一无二的，所以在 iOS 或 Mac App Store 中都不能为多个应用程序使用相同的 ID。这就是其格式以域名为基础的原因——如果我们拥有网站 usefulsoftware.com，那我们的所有捆绑包 ID 都将以 com.usefulsoftware 开头，这样就不会意外地使用别人正在使用或想要使用的捆绑包 ID，因为其他人都不会拥有同一域名。

如果没有域名，可以输入任何内容，只要它看起来像是一个反向的域名即可（比如 com.mycompany 就可以）。

如果计划发布你的 App，无论是发布到 App Store 还是其他什么地方，所用的公司标识符一定要与你拥有的域名相匹配，这一点非常重要。App Store 对此有强制要求，而且操作系统使用的捆绑包 ID 是根据这个公司标识符生成的，这就意味着在使用归自己所有的域名时，就不会在无意中创建一个与别人冲突的捆绑包 ID。

如果是为 Mac App Store 开发应用程序，还会提示输入 App Store 类别（是一个游戏、一个教育 App、一个社交网络 App，还是其他）。

根据选择的模板，可能还会向你询问其他一些信息（例如，如果正在创建一个可以识别文档的应用程序，比如 Mac App，会需要文档的文件扩展名）。还会询问你希望使用哪种语言；因为本书是关于 Swift 的，所以也许应当选择 Swift！这个项目会用到的其他信息将在后续步骤中补充。

(4) 为应用程序命名。在 Product Name 部分输入“HelloCocoa”。
(5) 让这个应用程序在 iPhone 上运行。从 Device（设备）下拉列表中选择 iPhone。

iOS 应用程序可以运行在 iPad 或 iPhone 上，也可以运行在这两种设备上。能够在这两种设备上运行的应用程序称为通用（universal）应用程序，它们运行相同的二进制文件，但拥有不同的用户界面。对于这个练习来说，选择 iPhone 即可。

(6) 单击 Next（下一步），创建项目。其他设置如图 1-2 所示。

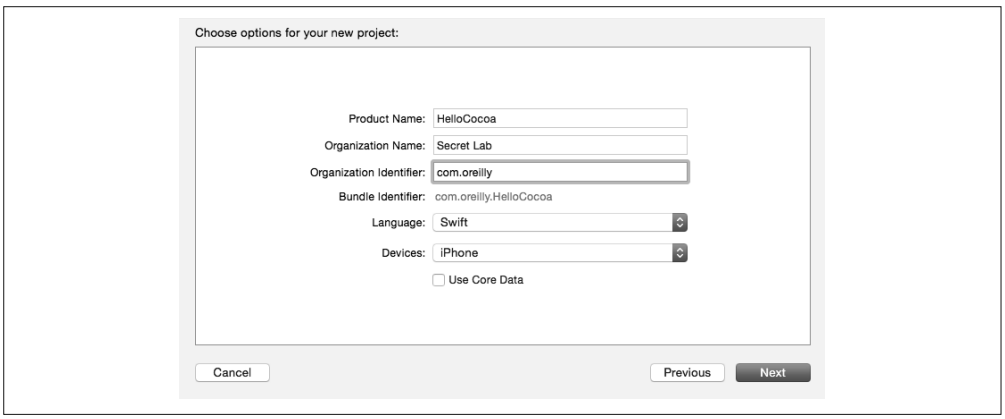

图 1-2：项目设置

(7) 选择项目的保存位置。这时会询问你要将项目保存到哪里。选择一个适当的位置。

完成上述步骤后，Xcode 将会打开该项目，现在可以开始使用整个 Xcode 界面了，如图 1-3 所示。

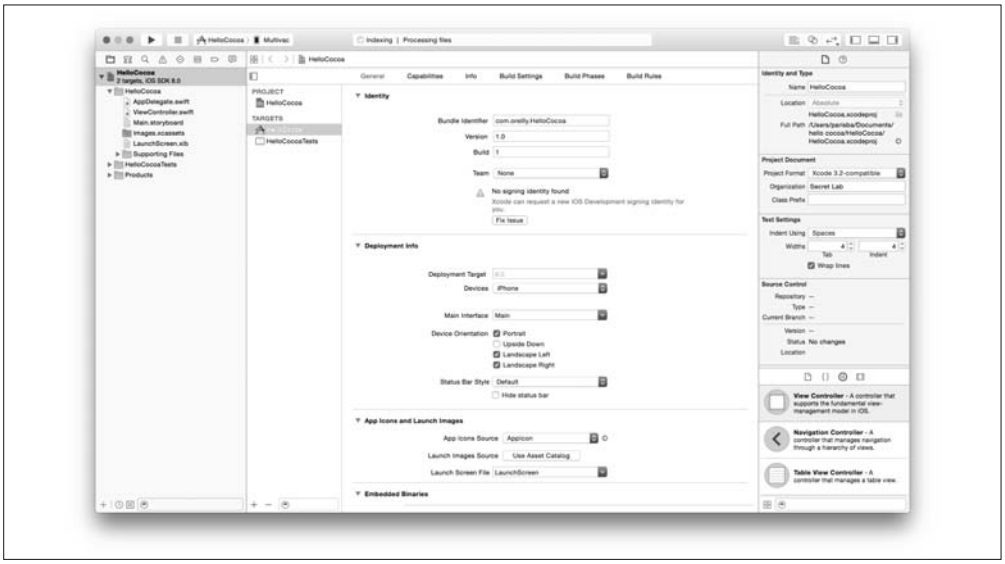

图 1-3：完整的 Xcode 界面

Xcode界面

前面曾经提到，Xcode 会在单个窗口中显示整个项目，而这个窗口会被分为许多分区。可以根据想要看到的内容，随意打开或关闭每个分区。

现在逐一了解一下这些分区，看看它们能做些什么。

1. 编辑器

Xcode 编辑器（如图 1-4 所示）是我们花费时间最多的地方。源代码编辑、界面设计和项目配置都在应用程序的这一分区内进行，它会根据当前打开的文件而发生变化。

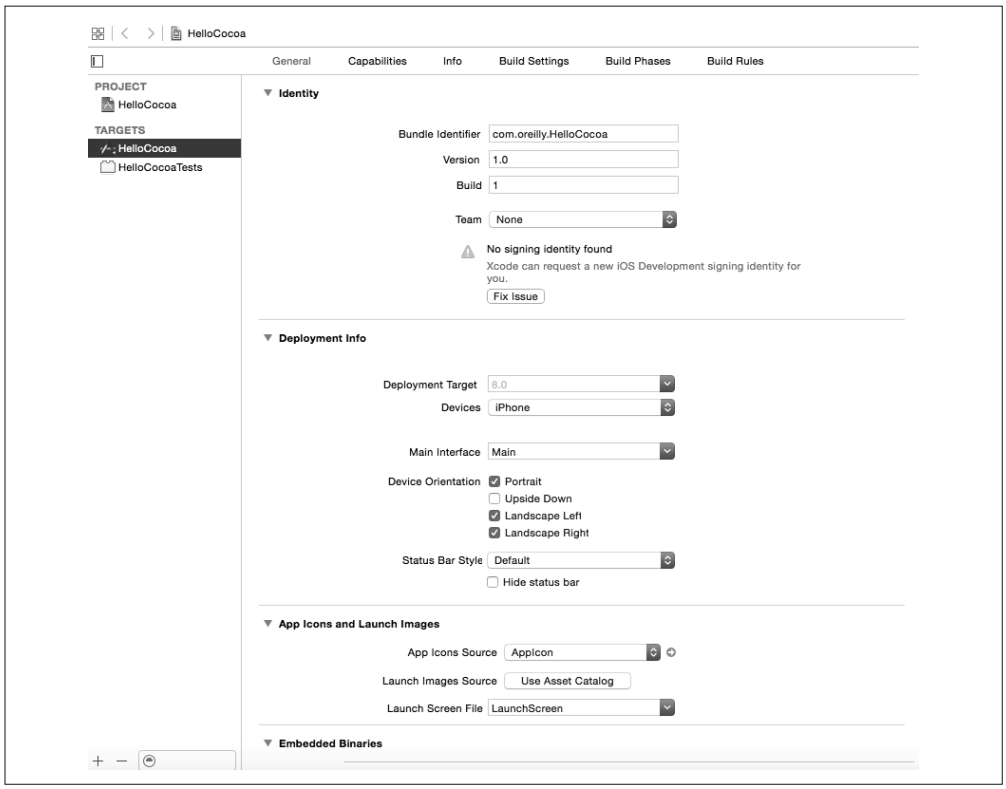

图 1-4：Xcode 的编辑器

如果正在编辑源代码，这个编辑器就是一个文本编辑器，具有代码自动完成、语法突出显示和开发人员希望集成开发环境拥有的所有有用功能。如果正在修改用户界面，这个编辑器就变成了一个可视化编辑器，可以拖动界面上的组件。其他类别的文件也有它们自己的专用编辑器。

这个编辑器还被划分为*主编辑器*和*助理编辑器*。当主编辑器中打开某个文件时，助理编辑器中会显示与该文件相关联的文件。即使主编辑器中打开了不同的文件，助理编辑器也会继续显示与当前所打开文件相关联的文件。

比如，在主编辑器中打开一个界面文件，然后打开助理编辑器，在默认情况下，助理编辑器会显示当前所编辑界面的相关代码。如果现在在主编辑器中打开另一个界面文件，助理编辑器中将显示新打开文件的相关代码。

我们还可以从编辑器中的一个文件直接跳到它的对应文件。例如，从界面文件跳到相应的实现文件。要进行这一跳转，只需按下 Control-Command- 上箭头键，在当前编辑器中打开当前文件的对应文件。还可以按下 Control-Command-Option- 上箭头键，在助理编辑器中打开当前文件的对应文件。

2. 工具栏

Xcode 工具栏（如图 1-5 所示）相当于整个界面的任务控件中心。在开发应用程序的过程中，它是 Xcode 中唯一不会发生显著变化的部分，我们可以在这里控制代码正在做的工作。

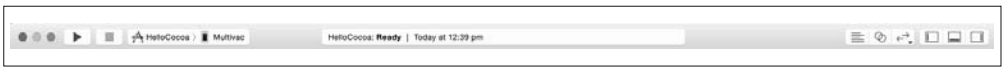

图 1-5：Xcode 的工具栏

在 OS X 窗口控件之后，这个工具栏自左向右的项目如下所示。

(1) Run（运行）按钮

单击这个按钮，将命令 Xcode 编译和运行应用程序。根据正在运行的应用程序种类和当前选择的设置，这个按钮将产生不同后果。

- 如果正在创建一个 Mac 应用程序，这个新 App 将出现在 Dock 中，并在机器上运行。
- 如果正在创建一个 iOS 应用程序，这个新 App 将在 iOS 模拟器或一个已连接的 iOS 设备（比如 iPhone 或 iPad）上运行。另外，如果单击此按钮，并按住鼠标左键不放，可以将它由“运行”改为其他操作，比如 Test（测试）、Profile（探查）或 Analyze（分析）。测试操作将运行任何已经设定的单元测试；探查操作将运行应用程序的 Instruments（见第 17 章）；分析操作将检查代码，指出潜在的问题和 bug。

(2) Stop（停止）按钮

单击这个按钮将停止 Xcode 正在执行的任何任务。如果它正在生成应用程序，将会停止生成；如果应用程序正在调试器中运行，将会退出。

(3) Scheme selector（方案选择器）

Xcode 将生成配置称为 scheme（方案）——也就是正在生成什么，如何生成。我们的项目可能包含多个目标（target），也就是应用程序创建的最终产品。目标之间可以共享资源，比如代码、声音和图片，从而可以更轻松地操控一项任务，比如生成一个 Mac 应用程序的 iOS 版本。我们不需要创建两个项目，而是创建一个具有两个目标的项目，这两个目标可以共享任意多的代码。要选择一个目标，可以单击方案选择器的左侧。我们还可以选择应用程序在哪里运行。如果正在生成一个 Mac 应用程序，那几乎总是希望在当前的 Mac 上运行它。但如果正在生成一个 iOS 应用程序，那就可以选择是在 iPhone 模拟器上运行这个应用程序，还是在 iPad 模拟器上运行。（它们实际上是同一个应用程序，只是根据内部运行的应用程序改变了形状。）如果已经连接了一个 iOS 设备，

并针对开发对其进行了设置，也可以选择在这个已连接设备上运行。

(4) Status display（状态显示）

状态显示部分显示了 Xcode 当前正在做的事情——生成应用程序、下载文档、在 iOS 设备上安装应用程序，等等。如果当前有多项任务同时进行，将会在左侧显示一个小按钮，单击该按钮将循环显示当前任务。

(5) Editor selector（编辑器选择器）

编辑器选择器决定了编辑器的布局。可以选择显示单个编辑器、带有助理编辑器的编辑器，或者是版本编辑器。如果正在使用诸如 Git 或 Subversion 之类的版本控制系统，则可以使用版本编辑器来比较一个文件的不同版本。

本书中几乎没有什么空间来讨论在项目中使用版本控制这个话题，但它的确是非常重要的。推荐阅读 Jon Loeliger 和 Matthew McCullough 的 *Version Control with Git, 2nd Edition*（O'Reilly)。

(6) View selector（视图选择器）

视图选择器控制着是否在屏幕上显示导航视图、调试视图和细节视图。如果屏幕空间紧张，或者只是希望降低一点混乱程度，只需单击这些元素，就能快速地显示和隐藏这些部分。

3. 导航器

Xcode 窗口的左侧是导航器（navigator)，其中给出了项目的相关信息（如图 1-6 所示)。

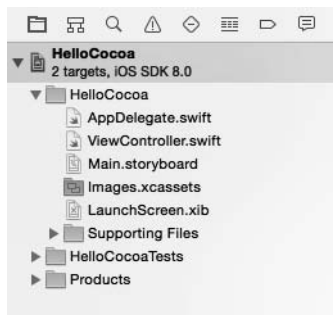

图 1-6：导航器窗格

导航器自左向右被分为八个选项卡。

- 项目导航器，给出一份列表，其中包括了组成该项目的所有文件。这是最常用的导航器，因为它决定了在编辑器中显示什么。项目导航器中选择的任何文件都会在编辑器中打开。

- 符号导航器，列出了项目中的所有类和函数。如果要查看一个类的概要，或者想直接跳转到这个类中的一个方法，那符号导航器是一个很便利的工具。
- 查找导航器，在整个项目中查找特定的文本。（快捷方式是 Command-Shift-F。按下 Command-F 将在当前打开的文档中进行查找。）
- 问题导航器，列出了 Xcode 在代码中发现的所有问题，其中包括了警告、编译错误，以及内建代码分析器发现的问题。
- 测试导航器，给出了与项目相关联的所有单元测试。单元测试曾经是 Xcode 的一个可选组件，但现在直接内建在 Xcode 中。单元测试将在 17.6 节讨论。
- 调试导航器，在调试程序时被激活，利用它可以查看程序各线程的状态。
- 断点导航器，列出了当前为进行调试而设置的所有断点。
- 报告导航器，列出了 Xcode 对项目完成的所有工作（比如，生成、调试和分析）。我们还可以回过头去查看之前 Xcode 会话中的生成报告。

4. 实用工具

实用工具窗格（如图 1-7 所示）显示了与编辑器中当前工作相关的一些补充信息。比如，如果正在编辑一个界面，实用工具窗格可以用来配置当前选择的用户界面元素。

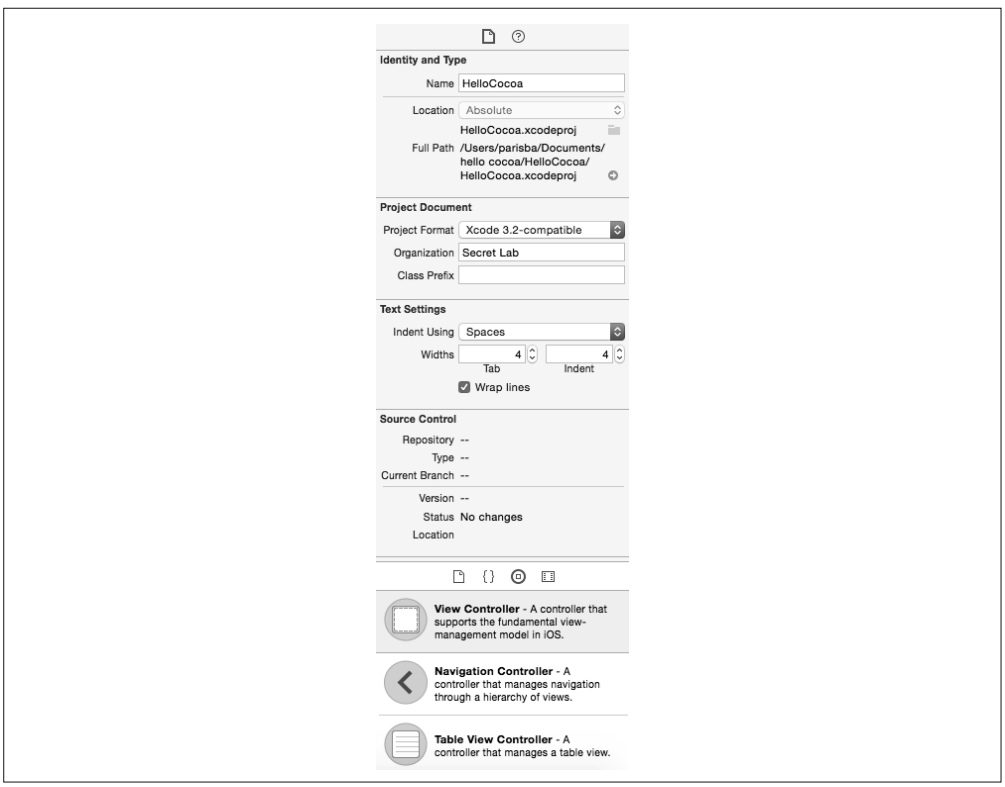

图 1-7：实用工具窗格

实用工具窗格分为两个部分：检查器，其中显示当前所选项目的其他细节和设置；库，这是一组可以添加到项目中的项（item）。在设计用户界面时会经常使用检查器和库；但是，库中还包含了很多有用的项，比如文件模板和代码片段，可以将它们拖放到位。

5. 调试分区

调试分区（如图 1-8 所示）显示了在运行程序时由调试器报告的信息。只要你希望查看应用程序在运行时会报告什么信息，就可以在调试分区查看它。

图 1-8：调试分区

这个分区被分为两部分：左侧在程序暂停时显示局部变量的值；右侧显示调试器正在生成的日志，其中包括来自被调试应用程序的所有日志信息。

1.3 开发一个简单的Swift应用程序

让我们开始使用 Xcode 吧。首先创建一个简单的 iOS 应用程序，然后将它连在一起。如果对 Mac 开发更感兴趣，不用担心，同样的技术仍然适用。

这个示例应用程序将显示单个按钮，在点击它时，会弹出一条警告，并将按钮的标签改为“Test!”。我们将继续使用前面创建的项目，所以请确保那个项目是打开的。

一般来说，先设计界面，然后再添加代码，这是一种好的程序设计实践。这意味着在编写代码时，已经理解了这些代码是如何与用户所见内容相对应的。

因此，我们将首先设计应用程序的界面。

1.3.1 设计界面

在使用 Cocoa 和 Cocoa Touch 生成应用程序界面时，有两个选项。一个选项是在故事板（storyboard）中设计应用程序的屏幕，故事板显示了所有屏幕是如何链接在一起的。另一个选项是分别设计各个屏幕。本书将在后面更详细地介绍故事板，就目前来说，我们编写的这个应用程序只有一个屏幕，所以选择哪种方式都没有太大关系。

首先打开界面文件，并添加一个按钮。下面是需要遵循的步骤。

(1) 首先打开故事板。因为新创建的项目在默认情况下使用故事板，所以这个 App 的界面存储在 Main.storyboard 文件中。

在项目导航器中选择该文件，将其打开。编辑器将会发生变化，显示此应用程序的单个、空白屏幕。

(2) 下面拖入一个按钮。接下来，向屏幕上增加一个按钮。所有用户界面控件都保存在对象库（object library）中，它在屏幕右侧实用工具窗格的底部。

为找到该按钮，可以在列表中滚动，直到找到 Button，或者在这个库底部的查找字段中键入“button”。

找到它之后，将它拖到屏幕中。

(3) 这时，需要对按钮进行配置。添加到界面中的每一项都是可以配置的。就目前来说，我们将只修改其标签。

单击这个新按钮，选中它，然后选择 Attributes Inspector（属性检查器），它是实用工具窗格顶端右数第三个选项卡。按下 Command-Option-4 按钮也可以找到它。

将这个按钮的 Title（标题）改为“Hello!”。

在界面中双击按钮也可以修改其标题。

这个简单的界面现在已经完成了（如图 1-9 所示）。唯一还没有完成的事情就是将它连接到代码。

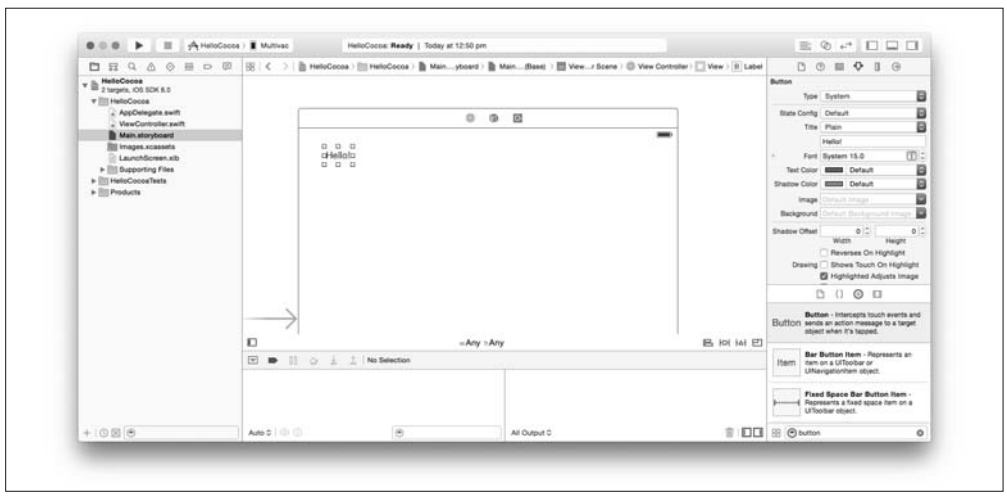

图 1-9：完成后的简单界面

1.3.2 连接代码

应用程序并不只是界面——作为开发者，还需要编写代码。为了使用前面设计的界面，需要在代码和界面之间建立连接。

我们可以建立两种连接。

- 输出口（outlet）是一些变量，用来引用界面中的对象。利用输出口可以命令一个按钮改变颜色或大小，或者隐藏它自己。Interface Builder 中还有一些输出口集合（outlet collection），可以用它来创建一组输出口，并选择其中包含哪些对象。
- 操作（action）是代码中的一些方法。当用户与对象交互时，将运行这些方法以作回应。这些交互包括用户用一根手指触碰一个对象、拖动一根手指，等等。

为使应用程序的行为与我们前面的描述保持一致——点击按钮显示标签，并改变按钮的文本，我们既要用到输出口，又要用到操作。操作将在单击按钮时运行，它使用连接到按钮的输出口来修改按钮标签。

要创建操作和输出口，需要同时打开界面编辑器和它的相应代码。然后按住 Control 键不放，将一个对象从界面编辑器拖到代码中（如果希望在界面中的两个对象之间建立连接，就将它拖到界面编辑器中的另一个对象上）。

现在将创建必要的连接。

(1) 首先，打开助理编辑器。为此，在工具栏中选择编辑器选择器中的第二个选项卡。

助理编辑器应当显示界面的相应代码 ViewController.swift。如果未显示，则单击交织在一起的圆圈图标（此图标表示助理编辑器），并导航到 Automatic → ViewController.swift。

如果正在使用 OS X 10.9 Mavericks，助理编辑器按钮看起来像是一件无尾礼服，而不是一对圆圈。

(2) 创建这个按钮的输出口。按住 Control 键不放，从按钮拖到代码中第一个“{”下方。

这时将出现一个弹出窗口。所有内容都保持默认设置，但将 Name 改为“helloButton”。单击 Connect。

这时将出现一行新代码：Xcode 已经为我们创建了连接，它作为类的一个属性出现在代码中。

(3) 创建这个按钮的操作。按住 Control 键不放，从按钮拖到刚刚创建的那行代码之下的空间中。此时将会再次出现一个弹出窗口。

这一次，将 Connection 由 Outlet 改为 Action。将 Name 设置为 `showAlert`。单击 Connect。这时将出现第二行新代码。Xcode 已经创建了连接，它是 `ViewController` 类内部的一个方法。

(4) 在刚刚创建的 `showAlert` 方法中，添加以下新代码：

```
@IBAction func showAlert(sender: AnyObject) {
    var alert = UIAlertController(title: "Hello!", message: "Hello, world!",
        preferredStyle: UIAlertControllerStyle.Alert)
    alert.addAction(UIAlertAction(title: "Close",
        style: UIAlertActionStyle.Default, handler: nil))
    self.presentViewController(alert, animated: true, completion: nil)
    self.helloButton.setTitle("Clicked", forState: UIControlState.Normal)
}
```

此代码创建了一个 `UIAlertController`，它在一个弹出窗口中向用户显示一条消息。为了做准备，它将窗口的标题改为“Hello!”，将窗口中的文本改为“Hello, world!”，然后向用户显示这一警示消息。最后，增加一个带有文本“Click”的操作（在本例中，它所做的就是让这条警示消息消失）。它随后将按钮的标题设定为“Clicked”。

应用程序现在已经做好运行的准备了。单击左上角的 Run 按钮。应用程序将会在 iPhone 模拟器中启动。

如果碰巧在计算机上连接了一个 iPhone 或 iPad，Xcode 在默认情况下将会尝试在该设备中启动此应用程序，而不是在模拟器中启动。为了让 Xcode 使用模拟器，可进入窗口左上角的 Scheme 菜单，将当前选择的方案改为模拟器。

当 App 在模拟器中完成启动时，点击按钮。这时将会出现一条警示消息；在关闭它时，会发现按钮的文本已经发生了变化。

1.4 使用iOS模拟器

利用 iOS 模拟器（如图 1-10 所示），无需使用实际设备就能测试 iOS 应用程序。这是一个有很用的工具，但一定要记住，模拟器的行为方式与实际设备有很大不同。

首先，模拟器的运行速度要比实际设备快得多，而且其内存也要大得多。这是因为模拟器利用的是计算机的资源，如果你的 Mac 有 8G RAM，那模拟器也将拥有这些内存；如果正在开发的应用程序需要处理器进行大量运算，那在模拟器上运行时要比在实际设备上顺畅得多。

iOS 模拟器可以模拟许多种不同的设备：从 iPad 2 到最新的 iPad，从 Retina 屏 3.5 英寸和 4 英寸的 iPhone 大小的设备，到最新的 4.7 英寸和 5.5 英寸的 iPhone，均可模拟。

图 1-10：iOS 模拟器

要改变设备，可打开 Hardware（硬件）菜单，选择 Device（设备），然后选择希望模拟的设备。还可以通过 Xcode 的方案选择器来改变要使用的模拟器。

我们还可以模拟硬件事件，比如 Home 按钮被按下，或者 iPhone 被锁定。要模拟按下 Home 按钮的操作，可以单击屏幕下方的虚拟按钮，也可以选择 Hardware → Home，还可以按下 Command-Shift-H。要锁定设备，可以按下 Command-L 或者选择 Hardware → Lock。

如果屏幕上没有空间，模拟器就不会显示虚拟硬件按钮。因此，如果你希望模拟按下 Home 按钮的操作，就需要使用快捷键，按下 Command-Shift-H。

模拟器中还有许多其他功能，我们在讨论 iOS 的各个部分时会一一涉及，届时会进行更详细的研究。

1.5 用TestFlight测试iOS App

TestFlight 是由苹果公司管理的一项服务，让你可以将 App 的副本发送给人们进行测试。利用 TestFlight，可以将 App 的生成版本发送给你所在机构的人，以及最多 1000 位外部测试者。

TestFlight 允许你将测试版本提交给“开发者计划”账户中的最多 25 位成员。另外，当 App 通过苹果公司的初步审核后，你可以另外将其发送给最多 1000 个人进行测试。

要使用 TestFlight，需要提供诸如 App 的名称、图标和说明等信息，在 iTunes Connect 中对应用程序进行配置。还要创建一份清单，列明应当接收该应用程序的用户。然后通过 Xcode 上载 App 的一个生成版本，苹果公司会通过电子邮件向这些用户发送一个链接，供他们下载和测试应用程序。

有关如何使用 TestFlight 的更多信息，请参阅 iTunes Connect 文档（https://developer.apple.com/library/ios/documentation/LanguagesUtilities/Conceptual/iTunesConnect_Guide/Chapters/BetaTestingTheApp.html#//apple_ref/doc/uid/TP40011225-CH35-SW3）。

O'REILLY®

Swift与Cocoa框架开发

Swift已经发布，你做好为iPhone、iPad和Mac开发应用的准备了吗？如果你是一位经验丰富的程序员，但从来没有接触过苹果的开发工具，本书会向你展示如何使用Cocoa和Cocoa Touch，用Swift语言开发出令人难以置信的iOS和OS X应用。

在本书中，你会学习如何在各种真实的环境中使用Swift，还有EventKit和Core Animation等Cocoa功能。在此过程中，你会了解Swift语言的功能和语法，理解为什么使用Swift比Objective-C开发iOS和Mac应用更容易、更快速、更安全。你还将完成几个练习，实践所学到的内容。

通过阅读本书，你将能够：

- **了解OS X和iOS应用的生命周期**
- **使用故事板设计自适应界面**
- **探索图形系统，包括内置的2D和3D游戏框架**
- **用AVFoundation显示视频和音频**
- **用文件系统在本地存储数据，或者用iCloud在网络上存储数据**
- **用表格视图和集合视图显示数据列表或数据集合**
- **开发可供用户创建、编辑和处理文档的应用**
- **使用MapKit、Core Location和Core Motion与现实世界交互**

Jonathon Manning

澳大利亚独立游戏开发工作室Secret Lab联合创始人，移动应用工程师、游戏设计师、程序员和计算学研究人员，从事过各种项目，曾为孩子们开发过iPad游戏，也开发过即时通信客户端。Twitter账号为@desplesda。

Paris Buttfield-Addison

Secret Lab联合创始人，移动应用工程师、游戏设计师和计算学研究人员，致力于使技术变得简单又有趣。Paris拥有计算学博士学位。Twitter账号为@parisba。

Tim Nugent

移动应用开发人员、游戏设计师、博士生、作家。大部分时间都用来设计和开发仅供自己娱乐的小应用和游戏。Twitter账号为@The_McJones。

MACINTOSH/IPAD & IPHONE

封面设计：Ellie Volckhausen　张健

图灵社区：iTuring.cn
热线：(010)51095186转600

分类建议　计算机/软件开发

人民邮电出版社网址：www.ptpress.com.cn

O'Reilly Media, Inc.授权人民邮电出版社出版

ISBN 978-7-115-39187-2

ISBN 978-7-115-39187-2
定价：89.00元