

iOS 9

畅销书全新升级, 累计印数60000册

- Swift和Objective-C双语讲解
- 新增通过代码构建界面、UI测试、AFNetworking和Alamofire、CocoaPods和Carthage等
- 数百个项目案例+两个真实项目开发全过程
- 涵盖iOS平台架构设计、测试驱动开发、性能优化、版本控制和程序调试等



iOS开发指南

从Hello World到App Store上架

(第4版)

关东升 著



中国工信出版集团



人民邮电出版社
POSTS & TELECOM PRESS

本书是iOS开发领域的权威指南，由苹果公司前首席工程师编写，内容全面，从Hello World到App Store上架，涵盖了iOS开发的方方面面。本书是iOS开发者的必读之作，也是iOS开发入门的首选教材。

本书是iOS开发领域的权威指南，由苹果公司前首席工程师编写，内容全面，从Hello World到App Store上架，涵盖了iOS开发的方方面面。本书是iOS开发者的必读之作，也是iOS开发入门的首选教材。

本书是iOS开发领域的权威指南，由苹果公司前首席工程师编写，内容全面，从Hello World到App Store上架，涵盖了iOS开发的方方面面。本书是iOS开发者的必读之作，也是iOS开发入门的首选教材。

iOS开发指南

从Hello World到App Store上架

(第4版)

关东升 著

人民邮电出版社

北京

图书在版编目 (C I P) 数据

iOS开发指南：从Hello World到App Store上架 /
关东升著. — 4版. — 北京：人民邮电出版社，2016.6
(图灵原创)
ISBN 978-7-115-42318-4

I. ①i… II. ①关… III. ①移动终端—应用程序—
程序设计 IV. ①TN929.53

中国版本图书馆CIP数据核字(2016)第095333号

内 容 提 要

本书是 iOS 开发权威指南，分 5 部分讲解如何从零起步编写并上线 iOS 应用。第一部分介绍 iOS 开发基础知识，包括界面构建技术、基本控件、协议、表视图、屏幕适配、导航、分屏多任务等。第二部分介绍设计与架构的相关知识，包括设计模式、分层模式、本地数据持久化等。第三部分为进阶篇，包括设置与配置、本地化、Contacts 与 ContactsUI 框架、数据交换格式、Web Service、定位服务、苹果地图等内容。第四部分介绍测试、调试和优化等相关知识。第五部分为实战篇，涵盖代码版本管理、项目依赖管理、App Store 发布流程，以及两个真实 iOS 应用的分析设计、编程、测试与发布过程。本书同时提供 Swift 和 Objective-C 这两种语言的代码。

本书适合所有 iOS 开发人员学习参考。

-
- ◆ 著 关东升
 - 责任编辑 王军花
 - 责任印制 彭志环
 - ◆ 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>
 - 北京艺辉印刷有限公司印刷
 - ◆ 开本：880×1230 1/16
 - 印张：51
 - 字数：1646千字 2016年6月第4版
 - 印数：56 001—61 000册 2016年6月北京第1次印刷
-

定价：119.00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：京东工商广字第 8052 号

前言

北京时间 2015 年 9 月 16 日凌晨，苹果公司正式推出了 iOS 9 系统。此次，Swift 2.0 也正式推出，Swift 经过一年的发展已经非常健壮了，很多项目也转而使用 Swift 语言开发，但是很多老项目还使用 Objective-C 语言开发。在这个大背景下，我们原来编写的《iOS 开发指南：从零基础到 App Store 上架》得到了广大读者的认可，并且很多读者希望我们将其升级为 iOS 9 版本。几个月过去了，我们终于在 2016 年春节之前将书稿提交给出版社。

内容和组织结构

本书是我们团队编写的 iOS 系列丛书中的一本，目的是使有 Swift 或 Objective-C 基础的程序员通过学习本书，从零基础学习如何在 App Store 上发布一款应用。全书共 5 部分。

第一部分为基础篇，共 9 章，介绍了 iOS 的一些基础知识。

第 1 章介绍了 iOS 的开发背景以及本书约定。

第 2 章使用故事板技术创建了 HelloWorld，同时讨论了 iOS 工程模板、应用的运行机制和生命周期、视图的生命周期等，最后介绍了如何使用 API 帮助文档和官方案例。

第 3 章重点介绍了 Cocoa Touch 框架中构建界面的相关类，重点介绍了构建界面的 3 种技术：故事板、XIB 和纯代码。

第 4 章重点介绍了标签、按钮、文本框、文本视图、开关、滑块、分段控件、网页控件、活动指示器、进度条、警告框、操作表、工具栏、导航栏等基本控件。而且，每个示例都采用故事板和纯代码这两种方式实现。

第 5 章介绍了数据源协议和委托协议，然后介绍了高级视图：选择器和集合视图。

第 6 章探讨了表视图的组成、表视图类的构成和表视图的分类，使我们对表视图有了一个整体上的认识。接下来，该章介绍了如何实现简单表视图和分节表视图，以及表视图中索引、搜索栏和分组的使用，然后学习了如何对表视图单元格进行删除、插入、移动等操作，最后介绍了表视图 UI 设计模式方面的内容。

第 7 章介绍了 iOS 多分辨率屏幕适配方法，其中涉及的技术主要是 Auto Layout 和 Size Class。此外，该章还介绍了 iOS 9 堆视图 StackView、iOS 屏幕的多样性、iOS 屏幕布局相关的内容。

第 8 章讨论了如何判断应用是否需要一个导航功能，并且知道在什么情况下选择平铺导航、标签导航、树形结构导航，或者同时综合使用这 3 种导航模式。

第 9 章首先介绍 iPhone 和 iPad 设备使用场景上的差异，然后介绍了 iPad 树形结构导航、iPad 模态视图和 Popover 视图，最后介绍了 iOS 9 分屏多任务。

第二部分为设计与架构篇，共 3 章，介绍了 iOS 设计与架构的相关知识。

第 10 章讨论了 iOS 开发中 4 种常用的设计模式，分别为单例模式、委托模式、观察者模式和 MVC 模式。在介绍每种设计模式时，我们按照问题提出、实现原理、应用案例的结构介绍了其适用情况、实现原理及用法。

第 11 章我们介绍了 iOS 平台的分层架构设计技术，归纳了 16 种分层模式，并且重点实现了 9 种模式，这些架构设计模式将贯穿全书（包括项目实战篇中的项目），希望读者能够重点学习。

第 12 章讨论了 iOS 本地数据持久化的问题。首先，该章分析了数据存取的几种方式以及每种数据存取方式适合什么样的场景，然后分别举例介绍了每种存取方式的实现。

第三部分为进阶篇，共7章，介绍了iOS的一些高级知识。

第13章介绍了设置和配置的概念，然后通过对二者差异的探讨，介绍了什么样的项目适合放在设置里，什么样的项目适合放在配置里。

第14章介绍了本地化的概念、内容和目录结构，并详细阐述了文本信息、XIB及故事板、资源文件的本地化。从技术角度看，应用程序的本地化比较简单，但是工作量大而且比较烦琐。

第15章介绍了iOS 9中新的访问通讯录框架——Contacts框架，然后介绍了如何使用该框架访问联系人信息、写入联系人信息等。最后，该章还介绍了如何使用ContactsUI框架提供系统界面，实现选择联系人、显示和修改联系人以及创建联系人的操作。

第16章重点介绍了数据交换格式，其中主要是介绍XML和JSON。

第17章介绍了Web Service，并重点讨论了REST风格Web Service，其中具体访问Web Service的API包括：NSURLSession、AFNetworking框架和Alamofire框架。

第18章讨论了iOS中的定位服务技术，包括地理信息编码和反编码查询。此外，该章还介绍了苹果的微定位技术iBeacon。

第19章讨论了iOS苹果地图的使用，包括显示地图、添加标注以及跟踪用户位置变化等。最后，我们介绍了程序外地图的使用。

第四部分为测试、调试和优化篇，共4章，介绍了iOS高级内容，包括测试、调试和优化等相关知识。

第20章首先介绍了iOS中都有哪些调试工具并重点介绍了几个常用的工具，具体包括日志与断言的输出、异常栈报告分析，接着讲解了如何在设备上调试应用，最后分析了Xcode设备管理工具的使用。

第21章讨论了测试驱动的iOS开发、测试驱动开发流程、单元测试框架XCTest，以及如何基于分层架构进行单元测试。

第22章首先介绍了iOS 9中提供的UI测试框架，最后介绍了如何基于分层架构进行UI测试。

第23章介绍了iOS中的性能优化方法，其中包括内存优化、资源文件优化、延迟加载、持久化优化、使用可重用对象、并发处理与多核CPU等。

第五部分为实战篇，共5章，首先介绍了iOS项目开发过程中的相关技术，包括代码版本管理、项目依赖管理以及App Store发布流程，最后从无到有地介绍了两个真实的iOS应用：MyNotes应用和2016里约热内卢奥运会应用。

第24章介绍了如何使用Git进行代码版本控制，其中包括Git服务器的搭建、Git常用命令和协同开发。此外，该章还介绍了在Xcode中如何配置和使用Git工具。

第25章讨论了iOS和OS X项目依赖管理工具，其中包括CocoaPods和Carthage。

第26章通过重构MyNotes应用，把MyNotes应用的数据由原来的本地存储变成云存储。在这个过程中，我们介绍了移动网络通信应用中分层架构设计的必要性和重要性。该章还重点介绍了基于委托模式和观察者模式通知机制实现的分层架构设计。

第27章探讨了如何在App Store上发布应用，介绍了应用的发布流程以及应用审核不通过的一些常见原因。

第28章介绍了完整的iOS应用分析设计、编程、测试和发布过程，其中采用了敏捷开发方法。此外，该项目采用分层架构设计，这对于学习iOS架构是非常重要的。

书中并没有包括多媒体等知识，我们会在另外一本iOS开发书中介绍，具体进展请读者关注智捷iOS课堂官方网站：<http://www.51work6.com>。

本书服务网址

为了更好地为广大读者提供服务，我们专门为本书建立了一个服务网址www.51work6.com/book/ios14.php，大家可以查看相关出版进度，并对书中内容发表评论，提出宝贵意见。

源代码

书中包括了 100 多个完整的案例项目源代码，大家可以到本书网站 www.51work6.com/book/ios14.php 下载或者到图灵社区（iTuring.cn）本书主页免费注册下载。

免费相关视频

为了使广大读者能够快速学习本书，智捷课堂给广大读者提供了此书相关的免费视频，具体包括：从零开始学 Swift、从零开始学 Objective-C 以及部分 iOS 相关视频。读者凭书中夹带的书签下方的优惠码，就可以购买价值 50 元内视频课程。具体使用说明：首先在智捷课堂视频平台（www.zhijieketang.com）上注册并登录，然后找到相应课程，接着选择页面中的“购买课程”→“去支付”→“输入优惠码”→“使用”即可。

勘误与支持

我们在网站 www.51work6.com/book/ios14.php 中建立了一个勘误专区，希望及时地把书中的问题、失误和纠正反馈给广大读者。如果你发现了任何问题，均可以在网上留言，也可以发送电子邮件到 eorient@sina.com，我们会在第一时间回复你。此外，你也可以通过新浪微博与我联系，我的微博为：@tony_关东升。

致谢

在此感谢图灵的王军花责编给我们提供的宝贵意见，感谢智捷 iOS 课堂团队的赵志荣参与内容讨论和审核，感谢赵大羽老师手绘了书中全部草图，并从专业的角度修改书中图片，力求更加真实完美地奉献给广大读者。此外，还要感谢我的家人容忍我的忙碌，以及对我的关心和照顾，使我能抽出这么多时间，投入全部精力专心编写此书。

由于时间仓促，书中难免存在不妥之处，请读者原谅。

关东升

2016 年 2 月于北京

目 录

第一部分 基础篇

第 1 章 开篇综述	2
1.1 iOS 概述	2
1.1.1 iOS 介绍	2
1.1.2 iOS 9 新特性	2
1.2 开发环境及开发工具	3
1.3 本书约定	4
1.3.1 案例代码约定	4
1.3.2 图示的约定	5
1.3.3 方法命名约定	6
1.3.4 构造函数命名约定	7
1.3.5 错误处理约定	8
第 2 章 第一个 iOS 应用程序	10
2.1 创建 HelloWorld 工程	10
2.1.1 创建工程	10
2.1.2 Xcode 中的 iOS 工程模板	14
2.1.3 应用剖析	15
2.2 应用生命周期	17
2.2.1 非运行状态——应用启动场景	19
2.2.2 点击 Home 键——应用退出场景	20
2.2.3 挂起重新运行场景	22
2.2.4 内存清除：应用终止场景	23
2.3 设置产品属性	23
2.3.1 Xcode 中的工程和目标	23
2.3.2 设置常用的产品属性	26
2.4 iOS API 简介	27
2.4.1 API 概述	27
2.4.2 如何使用 API 帮助	29
2.5 小结	32
第 3 章 Cocoa Touch 框架与构建应用界面	33
3.1 视图控制器	33
3.1.1 视图控制器种类	33
3.1.2 视图的生命周期	33

3.2 视图	35
3.2.1 UIView 继承层次结构	35
3.2.2 视图分类	37
3.2.3 应用界面构建层次	37
3.3 使用故事板构建界面	38
3.3.1 什么是故事板	39
3.3.2 场景和过渡	41
3.4 使用 XIB 文件构建界面	42
3.4.1 重构 HelloWorld	42
3.4.2 XIB 与故事板比较	45
3.5 使用纯代码构建界面	45
3.5.1 重构 HelloWorld	46
3.5.2 视图的几个重要属性	47
3.6 3 种构建界面技术讨论	49
3.6.1 所见即所得	49
3.6.2 原型驱动开发	49
3.6.3 团队协作开发	49
3.7 小结	50
第 4 章 UIView 与视图	51
4.1 标签与按钮	51
4.1.1 Interface Builder 实现	51
4.1.2 代码实现	55
4.2 事件处理	56
4.2.1 Interface Builder 实现	56
4.2.2 代码实现	59
4.3 访问视图	60
4.3.1 Interface Builder 实现	60
4.3.2 代码实现	62
4.4 TextField 和 TextView	63
4.4.1 Interface Builder 实现	64
4.4.2 代码实现	65
4.4.3 键盘的打开和关闭	67
4.4.4 关闭和打开键盘的通知	68
4.4.5 键盘的种类	69
4.5 开关控件、分段控件和滑块控件	70
4.5.1 开关控件	71

4.5.2 分段控件	72	6.3.2 代码实现	145
4.5.3 滑块控件	74	6.4 添加搜索栏	146
4.6 Web 视图: WKWebView 类	76	6.5 分节表视图	150
4.7 警告框和操作表	80	6.5.1 添加索引	150
4.7.1 UIAlertController 实现警告框	81	6.5.2 分组	153
4.7.2 UIAlertController 实现操作表	83	6.6 静态表与界面布局	154
4.8 等待相关的控件与进度条	84	6.7 插入和删除单元格	160
4.8.1 活动指示器 ActivityIndicatorView	85	6.7.1 Interface Builder 实现	162
4.8.2 进度条 ProgressView	87	6.7.2 代码实现	167
4.9 工具栏和导航栏	90	6.8 移动单元格	169
4.9.1 工具栏	90	6.9 表视图 UI 设计模式	171
4.9.2 导航栏	94	6.9.1 分页模式	171
4.10 小结	98	6.9.2 下拉刷新模式	171
6.9.3 下拉刷新控件	172	6.10 小结	174
第 5 章 委托协议、数据源协议与高级视图	99	第 7 章 界面布局与屏幕适配	175
5.1 视图中的委托协议和数据源协议	99	7.1 界面布局概述	175
5.2 选择器	99	7.2 iOS 界面布局设计模式	176
5.2.1 日期选择器	99	7.2.1 表单布局模式	176
5.2.2 普通选择器	104	7.2.2 列表布局模式	176
5.2.3 数据源协议与委托协议	109	7.2.3 网格布局模式	177
5.3 集合视图	111	7.3 传统布局技术	177
5.4 实例: Interface Builder 实现奥运会比赛 项目	112	7.4 Auto Layout 布局技术	179
5.4.1 添加集合视图	113	7.4.1 Interface Builder 中管理 Auto Layout 约束	179
5.4.2 添加集合视图单元格	114	7.4.2 实例: Auto Layout 布局	180
5.4.3 数据源协议与委托协议	118	7.5 iOS 9 堆视图 StackView	185
5.5 实例: 代码实现奥运会比赛项目	120	7.5.1 堆视图与布局	185
5.5.1 添加集合视图	120	7.5.2 实例: 堆视图布局	186
5.5.2 自定义集合视图单元格	122	7.6 iOS 屏幕的多样性	191
5.6 小结	123	7.6.1 iOS 屏幕介绍	191
第 6 章 表视图	124	7.6.2 iOS 的 3 种分辨率	192
6.1 概述	124	7.6.3 获得 iOS 设备屏幕信息	193
6.1.1 表视图的组成	124	7.7 Size Class 与 iOS 多屏幕适配	194
6.1.2 表视图的相关类	125	7.7.1 Interface Builder 中使用 Size Class	195
6.1.3 表视图分类	125	7.7.2 Size Class 的九宫格	195
6.1.4 单元格的组成和样式	127	7.7.3 实例: 使用 Size Class	196
6.1.5 数据源协议与委托协议	129	7.8 资源目录与图片资源适配	201
6.2 简单表视图	129	7.9 小结	204
6.2.1 实现协议方法	130	第 8 章 视图控制器与导航模式	205
6.2.2 UIViewController 根视图控制器	130	8.1 概述	205
6.2.3 UITableViewController 根视图 控制器	137	8.1.1 视图控制器的种类	205
6.3 自定义表视图单元格	140	8.1.2 导航模式	205
6.3.1 Interface Builder 实现	141		

8.2	模态视图	207
8.2.1	Interface Builder 实现	208
8.2.2	代码实现	214
8.3	平铺导航	215
8.3.1	应用场景	215
8.3.2	基于分屏导航的实现	217
8.3.3	基于电子书导航的实现	223
8.4	标签导航	228
8.4.1	应用场景	229
8.4.2	Interface Builder 实现	230
8.4.3	代码实现	234
8.5	树形结构导航	235
8.5.1	应用场景	235
8.5.2	Interface Builder 实现	236
8.5.3	代码实现	244
8.6	组合使用导航模式	245
8.6.1	应用场景	246
8.6.2	Interface Builder 实现	246
8.6.3	代码实现	254
8.7	小结	257

第9章 iPad 应用开发 258

9.1	iPad 与 iPhone 应用开发的差异	258
9.1.1	应用场景不同	258
9.1.2	导航模式不同	258
9.1.3	API 不同	258
9.2	iPad 树形结构导航	259
9.2.1	邮件应用中的树形结构导航	259
9.2.2	Master-Detail 应用程序模板	260
9.2.3	Interface Builder 实现 SplitViewSample 案例	266
9.2.4	代码实现 SplitViewSample 案例	270
9.3	iPad 模态视图	273
9.3.1	邮件应用中的模态导航	273
9.3.2	iPad 模态导航相关 API	275
9.3.3	Interface Builder 实现 ModalView-Sample 案例	277
9.3.4	代码实现 ModalViewSample 案例	281
9.4	Popover (浮动层) 视图	284
9.4.1	Popover 相关 API	284
9.4.2	PopoverViewSample 案例	285
9.5	iOS 9 分屏多任务	289
9.5.1	Slide Over 多任务	289
9.5.2	分屏视图多任务	290
9.5.3	画中画多任务	293

9.6	iPad 分屏多任务适配开发	293
9.6.1	分屏多任务前提条件	293
9.6.2	分屏多任务适配	294
9.7	小结	295

第二部分 设计与架构篇

第10章 iOS 常用设计模式 298

10.1	单例模式	298
10.1.1	问题提出	298
10.1.2	Objective-C 实现原理	298
10.1.3	Swift 实现原理	299
10.1.4	应用案例	300
10.2	委托模式	301
10.2.1	问题提出	301
10.2.2	实现原理	303
10.2.3	应用案例	306
10.3	观察者模式	309
10.3.1	问题提出	309
10.3.2	实现原理	310
10.3.3	通知机制和 KVO 机制	311
10.4	再谈 MVC 模式	317
10.4.1	MVC 模式概述	317
10.4.2	Cocoa Touch 中的 MVC 模式	318
10.5	小结	320

第11章 iOS 分层架构设计 321

11.1	低耦合企业级系统架构设计	321
11.2	iOS 分层架构设计	322
11.2.1	工程与工作空间	322
11.2.2	静态链接库	323
11.2.3	框架	323
11.2.4	16 种分层模式	324
11.3	实例: MyNotes 应用	325
11.4	基于同一工程的分层	327
11.4.1	采用纯 Swift 语言实现: PSSS 模式	327
11.4.2	采用纯 Objective-C 语言实现: POOO 模式	330
11.4.3	采用 Swift 调用 Objective-C 的混合搭配实现: PSOO 模式	334
11.5	基于同一工作空间静态链接库实现的 WLSOO 模式	335
11.5.1	使用工作空间	335
11.5.2	添加数据持久层工程	336

11.5.3	建立业务逻辑层与数据持久层依赖关系	337
11.5.4	建立表示层与业务逻辑层依赖关系	339
11.6	基于同一工作空间框架实现的 WFSSS 模式	339
11.6.1	创建框架工程	340
11.6.2	建立依赖关系	341
11.6.3	代码重构	342
11.7	基于同一工作空间框架实现的 WFOOO 模式	344
11.7.1	设置 Public 头文件	344
11.7.2	设置保护伞头文件	345
11.7.3	代码重构	345
11.8	基于同一工作空间框架实现的 WFSOO 模式	346
11.8.1	设置 Public 头文件	346
11.8.2	设置保护伞头文件	346
11.8.3	建立表示层与业务逻辑层依赖关系	346
11.8.4	代码重构	346
11.9	基于同一工作空间框架实现的 WFOSS 模式	346
11.9.1	设置 Public 头文件	346
11.9.2	设置 Swift 代码嵌入应用	347
11.9.3	代码重构	347
11.10	基于同一个工程不同目标框架实现的 TFSOO 模式	348
11.10.1	使用目标	348
11.10.2	添加框架目标	348
11.11	小结	349

第 12 章 数据持久化 350

12.1	概述	350
12.1.1	沙箱目录	350
12.1.2	持久化方式	351
12.2	属性列表	351
12.3	对象归档	357
12.4	使用 SQLite 数据库	361
12.4.1	SQLite 数据类型	361
12.4.2	配置 Objective-C 框架工程环境	362
12.4.3	配置 Swift 框架工程环境	362
12.4.4	创建数据库	364
12.4.5	查询数据	365
12.4.6	修改数据	368

12.5	Core Data	370
12.5.1	ORM	370
12.5.2	Core Data 栈	371
12.5.3	建模和生成实体	374
12.5.4	采用 Core Data 分层架构设计	380
12.5.5	查询数据	382
12.5.6	修改数据	384
12.6	小结	385

第三部分 进 阶 篇

第 13 章 应用程序设置 388

13.1	概述	388
13.1.1	设置	388
13.1.2	配置	389
13.2	应用程序设置包	390
13.3	设置项目种类	392
13.3.1	文本字段	395
13.3.2	开关	398
13.3.3	滑块	399
13.3.4	值列表	400
13.3.5	子界面	401
13.4	读取设置	403
13.5	小结	405

第 14 章 本地化 406

14.1	概述	406
14.1.1	需要本地化的内容	406
14.1.2	本地化目录结构	409
14.2	文本信息本地化	410
14.2.1	系统按钮和信息本地化	410
14.2.2	应用名称本地化	411
14.2.3	程序代码输出的静态文本本地化	413
14.2.4	使用 genstring 工具	414
14.3	故事板和 XIB 文件本地化	415
14.3.1	使用 Base Internationalization 技术	415
14.3.2	Auto Layout 与本地化	417
14.4	资源文件本地化	419
14.4.1	图片资源文件本地化	419
14.4.2	声音资源文件本地化	420
14.5	小结	421

第 15 章 iOS 9 中访问通讯录 422

15.1	通讯录的安全访问设置	422
15.2	使用 Contacts 框架读取联系人信息	423

15.2.1	查询联系人	423	17.4.4	实现 POST 请求	480
15.2.2	读取单值属性	426	17.4.5	下载数据	481
15.2.3	读取多值属性	428	17.4.6	上传数据	482
15.2.4	读取图片属性	430	17.5	使用为 Swift 设计的网络框架:	
15.3	使用 Contacts 框架写入联系人信息	430	Alamofire	484	
15.3.1	创建联系人	431	17.5.1	安装和配置 Alamofire 框架	484
15.3.2	修改联系人	433	17.5.2	实现 GET 请求	485
15.3.3	删除联系人	434	17.5.3	实现 POST 请求	486
15.4	使用系统提供界面	435	17.5.4	下载数据	487
15.4.1	选择联系人	435	17.5.5	上传数据	488
15.4.2	显示和修改联系人	438	17.6	反馈网络信息改善用户体验	489
15.4.3	创建联系人	441	17.6.1	使用下拉刷新控件改善用户体验	489
15.5	小结	444	17.6.2	使用活动指示器控件	492
17.6.3	使用网络活动指示器	494	17.7	小结	494
第 16 章	数据交换格式	445	第 18 章	定位服务	495
16.1	XML 数据交换格式	446	18.1	定位服务概述	495
16.1.1	XML 文档结构	446	18.1.1	定位服务编程	496
16.1.2	解析 XML 文档	447	18.1.2	测试定位服务	499
16.2	实例: MyNotes 应用 XML	448	18.2	管理定位服务	503
16.2.1	使用 NSXML 解析	450	18.2.1	应用启动与停止下的定位服务管理	504
16.2.2	使用 TBXML 解析	453	18.2.2	视图切换下的定位服务管理	504
16.3	JSON 数据交换格式	459	18.2.3	应用前后台切换下的定位服务管理	505
16.3.1	JSON 文档结构	459	18.2.4	设置自动暂停位置服务	507
16.3.2	JSON 数据编码/解码	460	18.2.5	iOS 9 后台位置服务管理	507
16.4	实例: MyNotes 应用 JSON 解码	461	18.3	地理信息编码与反编码	508
16.5	小结	462	18.3.1	地理信息反编码	508
第 17 章	REST Web Service	463	18.3.2	实例: 地理信息反编码	509
17.1	概述	463	18.3.3	地理信息编码查询	510
17.2	使用 NSURLSession	464	18.3.4	实例: 地理信息编码查询	510
17.2.1	NSURLSession API	464	18.4	微定位技术 iBeacon	512
17.2.2	简单会话实现 GET 请求	465	18.4.1	微定位与地理围栏	512
17.2.3	默认会话实现 GET 请求	468	18.4.2	iBeacon 技术概述	512
17.2.4	实现 POST 请求	469	18.4.3	实例: 使用 iBeacon 技术实现微定位	513
17.2.5	下载数据	470	18.5	小结	519
17.3	实例: 使用 NSURLSession 重构 MyNotes 案例	473	第 19 章	苹果地图应用	520
17.3.1	插入方法调用	473	19.1	使用 iOS 苹果地图	520
17.3.2	修改方法调用	474	19.1.1	显示地图	520
17.3.3	删除方法调用	475	19.1.2	显示 3D 地图	524
17.4	使用 AFNetworking 框架	477	19.2	添加标注	525
17.4.1	比较 ASIHTTPRequest、AFNetworking 和 MKNetworkKit	477			
17.4.2	安装和配置 AFNetworking 框架	478			
17.4.3	实现 GET 请求	479			

19.2.1	实现查询	526
19.2.2	在地图上添加标注	528
19.3	跟踪用户位置变化	529
19.4	使用程序外地图	530
19.5	小结	533

第四部分 测试、调试和优化篇

第20章 找出程序中的 bug——调试 536

20.1	Xcode 调试工具	536
20.1.1	定位编译错误	536
20.1.2	查看和显示日志	537
20.1.3	设置和查看断点	539
20.1.4	调试工具	544
20.1.5	输出窗口	545
20.1.6	变量查看窗口	546
20.1.7	查看线程	547
20.2	LLDB 调试工具	548
20.2.1	断点命令	548
20.2.2	观察点命令	550
20.2.3	查看变量和计算表达式命令	552
20.3	日志与断言输出	554
20.3.1	使用 NSLog 函数	554
20.3.2	使用断言	555
20.4	异常栈报告分析	556
20.4.1	跟踪异常栈	556
20.4.2	分析栈报告	558
20.5	在 iOS 设备上调试	559
20.5.1	Xcode 设置	559
20.5.2	设备设置	560
20.6	Xcode 设备管理工具	562
20.6.1	查看设备上的应用程序	562
20.6.2	设备日志	564
20.7	小结	564

第21章 iOS 测试驱动与单元测试 565

21.1	测试驱动的软件开发概述	565
21.1.1	测试驱动的软件开发流程	565
21.1.2	测试驱动的软件开发案例	566
21.1.3	iOS 单元测试框架	567
21.2	使用 XCTest 测试框架	568
21.2.1	添加 XCTest 到工程	568
21.2.2	Swift 版本中设置编译目标成员	570
21.2.3	编写 XCTest 测试方法	570
21.2.4	运行测试用例目标	573

21.2.5	分析测试报告	575
--------	--------	-----

21.3	异步单元测试	576
21.4	性能测试	580
21.4.1	Swift 版本中配置测试环境	580
21.4.2	测试用例代码	580
21.4.3	分析测试结果	581
21.5	iOS 单元测试最佳实践	583
21.5.1	配置测试数据持久层	583
21.5.2	编写数据持久层测试用例	584
21.5.3	运行测试数据持久层测试用例	587
21.5.4	配置测试业务逻辑层	588
21.5.5	编写业务逻辑层测试用例	588
21.6	小结	589

第22章 iOS 应用 UI 测试 590

22.1	UI 测试概述	590
22.2	添加 UI 测试到工程	590
22.2.1	创建工程时添加 UI 测试框架	590
22.2.2	在现有工程中添加 UI 测试用例 目标	591
22.3	录制脚本	593
22.3.1	录制之前的准备	593
22.3.2	录制过程	593
22.3.3	修改录制脚本	595
22.4	访问 UI 元素	595
22.4.1	UI 元素层次结构树	595
22.4.2	UI 测试中相关 API	597
22.5	表示层测试最佳实践	598
22.5.1	配置 UI 测试用例目标	598
22.5.2	编写 UI 测试用例	599
22.6	小结	602

第23章 让你的程序“飞”起来——性能优化 603

23.1	内存优化	603
23.1.1	内存管理	603
23.1.2	使用 Analyze 工具检查内存泄漏	603
23.1.3	使用 Instruments 工具检查内存 泄漏	608
23.1.4	使用 Instruments 工具检查僵尸 对象	612
23.1.5	autorelease 的使用问题	615
23.1.6	响应内存警告	616
23.2	优化资源文件	617
23.2.1	图片文件优化	618

23.2.2 音频文件优化	619
23.3 延迟加载	620
23.3.1 资源文件的延迟加载	620
23.3.2 故事板文件的延迟加载	624
23.3.3 XIB 文件的延迟加载	626
23.4 数据持久化的优化	628
23.4.1 使用文件	628
23.4.2 使用 SQLite 数据库	631
23.4.3 使用 Core Data	632
23.5 可重用对象的使用	634
23.5.1 表视图中的可重用对象	635
23.5.2 集合视图中的可重用对象	636
23.5.3 地图视图中的可重用对象	638
23.6 并发处理与多核 CPU	638
23.6.1 主线程阻塞问题	638
23.6.2 选择 NSThread 还是 GCD	639
23.7 小结	640

第五部分 实 战 篇

第 24 章 管理好你的程序代码——代码版本控制

24.1 概述	642
24.1.1 版本控制历史	642
24.1.2 基本概念	643
24.2 Git 代码版本控制	643
24.2.1 服务器搭建	643
24.2.2 Gitolite 服务器管理	645
24.2.3 Git 常用命令	647
24.2.4 Git 分支	649
24.2.5 Git 协同开发	653
24.2.6 Xcode 中 Git 的配置与使用	656
24.3 GitHub 代码托管服务	663
24.3.1 创建和配置 GitHub 账号	663
24.3.2 创建代码库	666
24.3.3 删除代码库	668
24.3.4 派生代码库	669
24.3.5 使用 GitHub 协同开发	671
24.3.6 管理组织	677
24.4 小结	680

第 25 章 项目依赖管理

25.1 使用 CocoaPods 工具管理依赖	681
25.1.1 安装 CocoaPods	681

25.1.2 搜索库	682
25.1.3 项目与第三方库搭配形式	683
25.1.4 示例：静态链接库形式管理依赖	684
25.1.5 示例：框架形式管理依赖	686
25.2 使用 Carthage 工具管理依赖	687
25.2.1 安装 Carthage	688
25.2.2 项目与第三方库搭配形式	688
25.2.3 Cartfile 文件	688
25.2.4 示例：重构 MyNotes 依赖关系	689
25.3 小结	691

第 26 章 重构 MyNotes 应用——iOS 网络通信中的设计模式与架构设计

26.1 移动网络通信应用的分层架构设计	692
26.2 Objective-C 版本：在数据持久层中添加和配置 AFNetworking	693
26.2.1 用 CocoaPods 工具管理依赖	693
26.2.2 测试依赖	693
26.3 Swift 版本：在数据持久层中添加和配置 Alamofire	694
26.3.1 用 Carthage 工具管理依赖	694
26.3.2 测试依赖	695
26.4 基于委托模式实现	695
26.4.1 网络通信与委托模式	696
26.4.2 在异步网络通信中使用委托模式实现分层架构设计	696
26.4.3 类图	697
26.4.4 时序图	699
26.4.5 数据持久层重构 (Objective-C 版本)	702
26.4.6 数据持久层重构 (Swift 版本)	705
26.4.7 业务逻辑层的代码实现	707
26.4.8 表示层的代码实现	710
26.5 基于观察者模式的通知机制实现	716
26.5.1 观察者模式的通知机制回顾	716
26.5.2 异步网络通信中通知机制的分层架构设计	716
26.5.3 类图	717
26.5.4 时序图	719
26.5.5 数据持久层重构 (Objective-C 版本)	722
26.5.6 数据持久层重构 (Swift 版本)	723
26.5.7 业务逻辑层的代码实现	725
26.5.8 表示层的代码实现	726
26.6 小结	731

第 27 章 把你的应用放到 App Store 上	732
27.1 收官	732
27.1.1 在 Xcode 中添加图标	732
27.1.2 在 Xcode 中添加启动界面	734
27.1.3 调整 Identity 和 Deployment Info 属性	738
27.2 为发布进行编译	739
27.2.1 创建开发者证书	739
27.2.2 创建 App ID	744
27.2.3 创建描述文件	745
27.2.4 发布编译	748
27.3 发布上架	750
27.3.1 创建应用	750
27.3.2 应用定价	753
27.3.3 基本信息输入	754
27.3.4 上传应用	757
27.3.5 提交审核	759
27.4 常见审核不通过的原因	761
27.4.1 功能问题	761
27.4.2 用户界面问题	761
27.4.3 商业问题	761
27.4.4 不当内容	761
27.4.5 其他问题	762
27.5 小结	762
第 28 章 iOS 敏捷开发项目实战——2016 里约热内卢奥运会应用开发及 App Store 发布	763
28.1 应用分析与设计	763
28.1.1 应用概述	763
28.1.2 需求分析	763
28.1.3 原型设计	764
28.1.4 数据库设计	765
28.1.5 架构设计	766
28.2 iOS 敏捷开发	766
28.2.1 敏捷开发宣言	766
28.2.2 iOS 适合敏捷开发?	767
28.2.3 iOS 敏捷开发最佳实践	767
28.3 任务 1: 创建应用工作空间和工程	769
28.3.1 迭代 1.1: 创建工作空间	769
28.3.2 迭代 1.2: 发布到 GitHub	769
28.4 任务 2: 信息系统层与持久层开发	770
28.4.1 迭代 2.1: 编写数据库 DDL 脚本	770

28.4.2 迭代 2.2: 插入初始数据到数据库	770
28.4.3 迭代 2.3: 数据库版本控制	771
28.4.4 迭代 2.4: 配置持久层工程 PersistenceLayer	771
28.4.5 迭代 2.5: 编写实体类	772
28.4.6 迭代 2.6: 编写 DAO 类单元测试用例类	773
28.4.7 迭代 2.7: 编写 DAO 类	775
28.4.8 迭代 2.8: 数据库帮助类 DBHelper	779
28.4.9 迭代 2.9: 配置及运行持久层测试用例目标	781
28.4.10 迭代 2.10: 发布到 GitHub	782
28.5 任务 3: 业务逻辑层开发	782
28.5.1 迭代 3.1: 比赛项目业务逻辑类 XCTest 单元测试	783
28.5.2 迭代 3.2: 编写比赛项目业务逻辑类	784
28.5.3 迭代 3.3: 比赛日程业务逻辑类 XCTest 单元测试	785
28.5.4 迭代 3.4: 编写比赛日程业务逻辑类	786
28.5.5 迭代 3.5: 发布到 GitHub	787
28.6 任务 4: 表示层开发	788
28.6.1 迭代 4.1: 使用资源目录管理图片和图标资源	788
28.6.2 迭代 4.2: 根据原型设计初步设计故事板	789
28.6.3 迭代 4.3: “首页”模块	790
28.6.4 迭代 4.4: “比赛项目”模块	791
28.6.5 迭代 4.5: “比赛日程”模块	795
28.6.6 迭代 4.6: “倒计时”模块表示层	797
28.6.7 迭代 4.7: “关于我们”模块表示层	800
28.6.8 迭代 4.8: 发布到 GitHub	800
28.7 任务 5: 收工	800
28.7.1 迭代 5.1: 添加图标	800
28.7.2 迭代 5.2: 设计和添加启动界面	801
28.7.3 迭代 5.3: 性能测试与改善	801
28.7.4 迭代 5.4: 发布到 GitHub 上	802
28.7.5 迭代 5.5: 在 App Store 上发布应用	802
28.8 小结	802

Part 1

第一部分

基础篇

本 部 分 内 容

- 第 1 章 开篇综述
- 第 2 章 第一个 iOS 应用程序
- 第 3 章 Cocoa Touch 框架与构建应用界面
- 第 4 章 UIView 与视图
- 第 5 章 委托协议、数据源协议与高级视图
- 第 6 章 表视图
- 第 7 章 界面布局与屏幕适配
- 第 8 章 视图控制器与导航模式
- 第 9 章 iPad 应用开发

自从App Store上线以来，它创造了很多神话，给我们这些程序员提供了展示自己的舞台，给了我们创意的空间，给了我们创业的机会。下面让我们从这里开始iOS开发之旅吧。

1.1 iOS 概述

在本节中，我们将了解什么是iOS以及iOS 9有哪些新特性。

1.1.1 iOS 介绍

iOS是由苹果公司开发的移动设备操作系统，这些移动设备包括iPhone、iPod touch和iPad等，目前最新的操作系统是iOS 9。

苹果公司最早于2007年1月9日的Macworld大会上公布了这个系统，最初是设计给iPhone使用的，后来陆续被应用到iPod touch和iPad等产品上。iOS与苹果的Mac OS X操作系统一样，都属于类Unix的商业操作系统。

原本这个系统的名字为iPhone OS，因为主要应用于iPhone和iPod touch设备，后来在2010 WWDC大会上宣布改名为iOS。

1.1.2 iOS 9 新特性

iOS的最新版本为iOS 9.3。苹果公司于2015年9月18日凌晨1点开放其正式版的下载，它支持iPhone 4s、iPhone 5、iPhone 5c、iPhone 5s、iPhone 6、iPhone 6 Plus、iPhone 6s、iPhone 6s Plus、iPod touch 5、iPod touch 6、iPad 2、iPad (第三代)、iPad (第四代)、iPad Air、iPad Air 2、iPad mini、iPad mini 2、iPad mini 3、iPad mini 4和iPad Pro等设备。据苹果发布的更新文档显示，iOS 9增强了对CloudKit、HomeKit、HealthKit和MapKit等已有技术的支持，另外，iOS 9新增了多项功能，很多新特性或将成为将来的焦点。

现在我们先简要介绍一下iOS 9几个重要的变化。

- ❑ **3D Touch**。为安装iOS 9的iPhone 6s和iPhone 6s Plus设备提供了一种新的交互方式。用户轻按屏幕会弹出菜单。
- ❑ **分屏多任务**。iOS 9提供了对iPad设备的分屏多任务支持，可以在同一个屏幕上同时显示运行中的两个应用，我们可以一边看视频一边发微信。iPad分屏多任务功能分为3个形式：Slide Over、Split View（分屏视图）和画中画（Picture in Picture）。
- ❑ **Apple Pay 增强**。iOS 9中增强了Apple Pay功能，它是苹果的手机支付功能，苹果一直看好手机支付领域，目前Apple Pay已经进入中国市场。
- ❑ **Wallet**。iOS 9添加了对商店信用卡、奖励卡以及会员卡的支持。原来的Passbook 功能被改名为Wallet。此外，iOS 9还对已有的框架进行了不同程度的增强和删减。

1.2 开发环境及开发工具

苹果公司于2008年3月6日发布了iPhone和iPod touch的应用程序开发包,其中包括Xcode开发工具、iPhone SDK和iPhone手机模拟器。第一个Beta版本是iPhone SDK 1.2b1 (build 5A147p),发布后立即就能使用,但是同时推出的App Store所需要的固件更新直到2008年7月11日才发布。我们编写本书时,iOS SDK 9.3版本已经发布。

iOS开发工具主要是Xcode。自从Xcode 3.1发布以后,Xcode就成为iPhone软件开发工具包的开发环境。Xcode可以开发Mac OS X和iOS应用程序,其版本是与SDK相对应的。例如,Xcode 6与iOS SDK 8对应、Xcode 7与iOS SDK 9对应。

在Xcode 4.1之前,还有一个配套使用的工具Interface Builder,它是Xcode套件的一部分,用来设计窗体和视图,可用于“所见即所得”地拖曳控件并定义事件等,其数据以XML的形式被存储在XIB文件中。在Xcode 4.1之后,Interface Builder成为了Xcode的一部分,与Xcode集成在一起。

打开Xcode 7工具,看到的主界面如图1-1所示。该界面主要分成5个区域:①号区域是菜单栏,其中的菜单几乎包括Xcode的所有功能;②号区域是工具栏,其中的按钮可以完成大部分工作;③号区域是导航面板,主要是对工作空间中的内容进行导航;④号区域是代码编辑区,我们的编码工作就是在这里完成的;⑤号区域是检查器,包括:文件检查器、属性检查器、标识检查器和尺寸检查器等。

在导航面板上面还有一排按钮,如图1-2所示,默认选中的是“文件”导航面板。关于各个按钮的具体用法,我们会在以后用到的时候详细介绍。

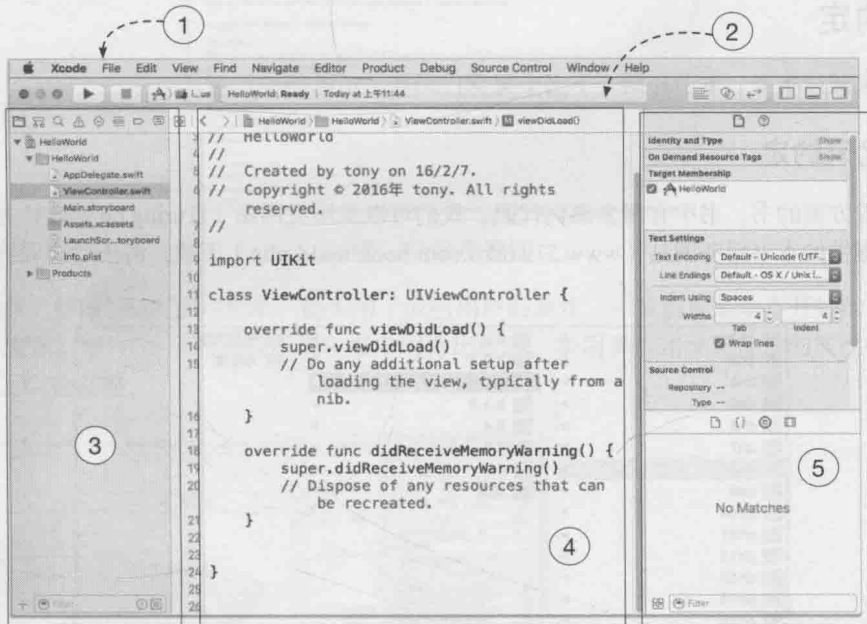


图1-1 Xcode主界面

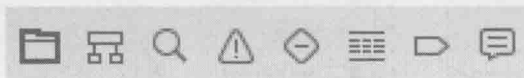


图1-2 Xcode导航面板

在选中导航面板时,导航栏下面也有一排按钮,如图1-3所示。这是辅助按钮,它们的功能都与该导航面板的内容相关。对于不同的导航面板,这些按钮也是不同的。