第一图书网, tushu007.com

<<iOS 6编程实战>>

图书基本信息

书名：<<iOS 6编程实战>>

13位ISBN编号：9787115312181

10位ISBN编号：7115312184

出版时间：2013-4

出版时间：人民邮电出版社

作者：[美] Rob Napier,[美] Mugunth Kumar

译者：陈晓亮,武海峰,邓　强,周庆成

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：http://www.tushu007.com

前言

能生活在这个时代，既不幸，又幸运。
不幸的是，百年之后人类的生活将是现在的我们所无法想象的。
幸运的是，世界在飞速变化，改变世界的产品在不断出现。
今天的移动互联网就是改变人类生活的强大力量。
哪怕是10年前，正当互联网如火如荼时，我们也不会想到在今天智能移动设备会如此普及，人们能无时无刻地访问网络。
而在所有的移动操作系统中，iOS可能不是市场占有率最高的，却无疑是最优秀的。
无论是在UI设计、交互设计还是应用质量上，iOS应用总体上来讲都要优于其他移动平台的应用，这固然是因为iOS开发者整体上比较优秀，同样也和iOS本身的高质量SDK分不开。
然而，要成为优秀的iOS开发者，却并不是容易的事。
我一直认为苹果的官方文档是最好的学习资料，但是本书确实质量很高，两位作者凭借多年的实践经验，向我们展示了从初级到高级的各种iOS编程技巧和理念，相信会对读者有所助益。
全书的翻译由陈晓亮、武海峰、邓强、周庆成完成，由于时间仓促（真的是时间仓促，不是套话），尽管我们努力地统一术语，检查错误，但是肯定会有所疏漏，如果读者发现有错误，我们欢迎拍砖，同时也欢迎来信交流。

内容概要

《图灵程序设计丛书:iOS 6编程实战》急开发者之所急，揭秘了多数开发类图书未曾展示过的iOS 6高级特性和开发技巧，带你深入了解iOS 6。
作者结合自己多年的实践经验，毫无保留、面面俱到地讲解了如何充分利用强大的SDK将你的移动开发技能发挥到极致。
全书涵盖了安全、多任务处理、多平台运行、块与函数式编程、高级文本布局、性能调优，以及如何通过应用内购买功能提高销售业绩。

《图灵程序设计丛书:iOS 6编程实战》精彩收录：兼容iPhone 5 充分利用集合视图；操作Objective-C运行时 性能调优和消除BUG；用Security Services确保数据安全 面向REST式Web服务创建高性能应用；实现诱人的动画效果 在网络连接质量不一的情况下实现超快缓存；用Core Text打造精妙的文本布局 充分发挥Core Foundation的强大功能。

作者简介

作者：（美国）Rob Napier （美国）Mugunth Kumar 译者：陈晓亮　武海峰　邓强　周庆成Rob Napier：2005年开始从事Mac开发，iPhone SDK第一版发布时就开始开发iPhone应用。
他是The Daily、PandoraBoy和Cisco Mobile的作者，Stack Overflow的主要贡献者，并且维护着技术博客Cocoaphony（cocoaphony.com），主要关注组网、性能、安全和MVC模式。
他自建了三套房子，喜欢徒步旅行，还是一位自豪的父亲。
Mugunth Kumar：新加坡独立iOS开发者，拥有新加坡南洋理工大学硕士学位，主修信息系统。
其个人博客（blog.mugunthkumar.com）主要发表移动开发、软件可用性、用户体验和iOS相关的文章。
从事iOS开发前，他在财富500强企业通用电气和霍尼韦尔担任Windows和.NET平台软件顾问。
他关注的领域包括编程方法论（面向对象和函数式编程）、移动开发以及可用性设计。
不写代码的时候，他会到国外拍摄一些大自然风景。
陈晓亮：iOS开发工程师，主要关注Linux、Mac和iOS，喜欢Lisp，也喜欢Objective-C。
武海峰：Linux系统工程师，主要关注GNU/Linux和移动互联应用，热衷于开源软件。
曾在商业Linux厂商和移动互联网Startup任职，从事Android系统集成、商业Linux发行版集成及移动互联网产品开发。
邓强：高级软件工程师，关注互联网行业，对互联网产品和技术有深厚兴趣，Apple粉，Google粉，Amazon粉。
目前就职于美国道富银行杭州研发中心，从事金融行业软件系统开发。
周庆成：互联网与移动应用开发者，对iOS等移动系统拥有极大兴趣，开发过iPad版三人斗地主等游戏，且对于各种系统平台与编程语言都有研究。
此外，他还翻译了《Objective-C基础教程（第2版）》。
目前居于上海，从事游戏与网络应用开发。

书籍目录

第一部分 全新功能 第1章 全新的系统 2 1.1 新功能 2 1.1.1 集合视图 3 1.1.2 自动布局 3 1.1.3 自动引用计数 3 1.1.4 应用内购买的托管内容 4 1.1.5 社交框架 4 1.1.6 UI状态保留 4 1.1.7 其他新功能 4 1.2 小结 6 1.3 扩展阅读 6 第2章 熟悉Xcode4和LLVM编译器 7 2.1 了解用户界面 8 2.1.1 标签式编辑器 9 2.1.2 按键绑定的变化 9 2.1.3 工程设置编辑器 10 2.1.4 内置的版本控制系统 10 2.1.5 工作区 10 2.2 单窗口工作方式 10 2.3 导航面板 11 2.3.1 工程导航面板 12 2.3.2 符号导航面板 12 2.3.3 搜索导航面板 12 2.3.4 问题导航面板 12 2.3.5 调试导航面板 12 2.3.6 断点导航面板 12 2.3.7 日志导航面板 13 2.4 辅助功能 13 2.5 内置的InterfaceBuilder 13 2.6 Xcode之核心：LLVM编译器 14 2.6.1 Clang编译器前端 15 2.6.2 我是一个Bug！
修复我 15 2.7 LLVM 4的新功能 15 2.7.1 字面量 15 2.7.2 字面量和可用性检查 18 2.7.3 实例变量的自动合成 18 2.8 使用Git进行版本控制 19 2.8.1 内置的Git版本控制系统 19 2.8.2 版本编辑器 19 2.8.3 Git最佳实践 19 2.9 Scheme 20 2.9.1 为什么使用Scheme 20 2.9.2 建议用Scheme实现你的意图 21 2.9.3 创建一个Scheme 21 2.9.4 共享你的Scheme 22 2.10 为编译配置添加注释 22 2.10.1 创建xcconfig文件 22 2.10.2 重构编译配置文件 23 2.11 将代码转换为支持ARC的Objective—C代码 23 2.12 Xcode4Organizer 25 2.12.1 AutomaticDeviceProvisioning 25 2.12.2 查看崩溃日志和控制台输出 25 2.12.3 查看应用沙盒数据 26 2.12.4 管理版本库 26 2.12.5 管理应用程序归档 26 2.12.6 查看Objective—C和SDK文档 26 2.13 小结 26 2.14 扩展阅读 26 2.14.1 苹果文档 26 2.14.2 WWDC讲座 27 2.14.3 博客 27 2.14.4 网上资源 27 2.14.5 图书 27 第二部分 熟能生巧 第3章 Objective—C基础知识 30 3.1 命名约定 30 3.2 自动引用计数 32 3.3 属性 35 3.3.1 属性修饰关键字 36 3.3.2 属性最佳实践 37 3.3.3 私有实例变量 37 3.4 存取器 37 3.5 分类和扩展 38 3.5.1 +load方法 40 3.5.2 使用关联引用为分类添加数据 40 3.5.3 类扩展 42 3.6 正式协议和非正式协议 42 3.7 小结 43 3.8 扩展阅读 44 3.8.1 苹果文档 44 3.8.2 其他资源 44 第4章 Cocoa设计模式 45 4.1 理解模型？
视图？
控制器 45 4.1.1 使用模型类 46 4.1.2 使用视图类 46 4.1.3 使用控制器类 47 4.2 理解委托和数据源 47 4.3 使用命令模式 48 4.3.1 使用目标？
动作 48 4.3.2 使用方法签名和调用 49 4.3.3 使用蹦床 52 4.3.4 撤销 55 4.4 使用观察者模式 55 4.5 使用单例模式 58 4.6 小结 60 4.7 扩展阅读 60 4.7.1 苹果文档 60 4.7.2 其他资源 61 第5章 通过Objective—C的ARC管理内存 62 5.1 Objective—C的ARC基础入门 62 5.1.1 历史简介 62 5.1.2 手动引用计数与自动引用计数 63 5.2 什么是ARC 63 5.2.1 在ARC工程中集成非ARC的第三方代码 64 5.2.2 在非ARC工程中集成ARC代码 64 5.2.3 框架中的ARC代码 64 5.2.4 ARC空声明变量 67 5.2.5 Objective—C命名约定 67 5.2.6 覆盖默认行为 67 5.2.7 自由桥接 68 5.2.8 ARC内部原理 68 5.3 常见的ARC移植错误 69 5.4 小结 72 5.5 扩展阅读 72 5.5.1 苹果文档 72 5.5.2 WWDC讲座 72 5.5.3 博客 73 第6章 熟练使用表视图 74 6.1 UITableView类的继承关系 74 6.2 熟悉表视图 75 6.2.1 UITableViewController 75 6.2.2 UITableViewCell 75 6.2.3 提高表视图的滚动速度 75 6.2.4 自定义非重复表单元 82 6.2.5 表视图最佳方案：编写简洁代码，简化控制器内容 83 6.3 小结 87 6.4 扩展阅读 87 6.4.1 苹果文档 87 6.4.2 其他资源 87 第7章 集合视图与自动布局 88 7.1 集合视图 88 7.1.1 类与协议 88 7.1.2 示例 89 7.2 自动布局 97 7.2.1 使用自动布局 98 7.2.2 了解自动布局 98 7.2.3 相对布局约束 98 7.2.4 视觉化格式语言 101 7.2.5 调试布局错误 102 7.3 小结 103 7.4 扩展阅读 104 7.14.1 苹果文档 104 7.14.2 WWDC讲座 104 第8章 绘图功能 105 8.1 iOS的不同绘图系统 105 8.2 UIKit和视图绘图周期 106 8.3 视图绘制与视图布局 108 8.4 自定义视图绘制 108 8.4.1 通过UIKit绘图 109 8.4.2 路径 109 8.4.3 理解坐标系 111 8.4.4 重新调整大小以及内容模式 114 8.4.5 变形 115 8.4.6 通过CoreGraphics进行绘制 117 8.4.7 混用UIKit与CoreGraphics 120 8.4.8 管理图形上下文 121 8.5 优化UIView绘制 123 8.5.1 避免绘图 123 8.5.2 缓存与后台绘制 123 8.5.3 自定义绘图与预渲染 123 8.5.4 像素对齐与模糊文本 124 8.5.5 透明、不透明与隐藏 125 8.6 CGLayer 125 8.7 小结 127 8.8 扩展阅读 127 8.8.1 苹果文档 127 8.8.2 其他资源 128 第9章 CoreAnimation 129 9.1 视图动画 129 9.2 管理用户交互 131 9.3 图层绘制 132 9.3.1 直接设置内容 134 9.3.2 实现display方法 134 9.3.3 自定义绘图 135 9.3.4 在自己的上下文中绘图 136 9.4 移动对象 136 9.4.1 隐式动画 137 9.4.2 显式动画 137 9.4.3 模型与表示 138 9.4.4 关于定时 140 9.5 三维动画 142 9.6 美化图层 145 9.7 用动作实现自动动画 146 9.8 为自定义属性添加动画 147 9.9 CoreAnimation与线程 148 9.10 小结 149 9.11 扩展阅读 149 9.11.1 苹果文档 149 9.11.2 其他资源 149 第10章 错误处理 150 10.1 错误处理模式 150 10.2 断言 151 10.3 异常 153

10.4 捕获与报告程序崩溃 153 10.5 错误和NSError 154 10.6 错误处理块 156 10.7 日志 157 10.7.1 记录敏感信息 158 10.7.2 获取日志文件 159 10.8 小结 159 10.9 扩展阅读 159 10.9.1 苹果文档 159 10.9.2 其他资源 159 第11章 位置服务 161 11.1 CoreLocation框架 161 11.1.1 获取用户位置 162 11.1.2 使用内置的罗盘获取用户前进方向 162 11.1.3 位置服务和用户隐私 163 11.2 后台位置服务 165 11.2.1 后台获取位置更新 165 11.2.2 显著位置变更通知 165 11.2.3 区域监控（地理围栏） 167 11.3 位置模拟 167 11.4 时刻关注电量消耗 168 11.5 小结 168 11.6 扩展阅读 168 第三部分 选择工具 第12章 表视图常用UI范例 170 12.1 下拉刷新 170 12.2 无限滚动 173 12.3 行内编辑与键盘 175 12.4 UITableView的动画效果 177 12.5 表视图动画的实际应用 178 12.5.1 实现折叠列表 178 12.5.2 实现选项抽屉 180 12.5.3 在表视图单元格中使用手势识别器 181 12.6 小结 181 12.7 扩展阅读 182 第13章 多任务 183 13.1 后台运行最佳实践：能力越大责任越大 183 13.2 状态恢复系统 185 13.2.1 测试状态恢复系统 186 13.2.2 选择性加入 186 13.2.3 应用启动过程的变化 187 13.2.4 状态恢复标识符 187 13.2.5 状态编码器与状态解码器 188 13.2.6 表视图和集合视图 191 13.2.7 状态恢复系统的高级应用 192 13.3 多任务和运行循环简介 192 13.4 以操作为中心的多任务开发 193 13.5 用GCD实现多任务 197 13.5.1 GCD和ARC 198 13.5.2 源和定时器 198 13.5.3 用分派屏障创建同步点 199 13.5.4 队列目标和优先级 200 13.5.5 分派组 201 13.5.6 队列关联数据 202 13.5.7 分派数据 202 13.6 小结 203 13.7 扩展阅读 203 13.7.1 苹果文档 203 13.7.2 WWDC讲座 203 13.7.3 其他资源 203 第14章 REST解惑 204 14.1 REST简介 205 14.2 选择数据交换格式 205 14.2.1 在iOS中解析XML 205 14.2.2 在iOS中解析JSON 206 14.2.3 XML与JSON 207 14.2.4 模型版本化 208 14.3 假想的Web服务 208 14.4 重要提醒 209 14.5 RESTfulEngine架构（iHotelApp示例代码） 209 14.5.1 NSURLConnection与第三方框架 209 14.5.2 创建RESTfulEngine 210 14.5.3 使用访问令牌对API调用进行认证 213 14.5.4 在RESTfulEngine.m中覆盖相关方法以添加自定义认证头部 213 14.5.5 取消请求 214 14.5.6 请求响应 214 14.5.7 对JSON数据进行键值编码 215 14.5.8 列表页面的JSON对象与详细页面的JSON对象 217 14.5.9 嵌套JSON对象 217 14.5.10 少即是多 219 14.5.11 错误处理 219 14.5.12 本地化 221 14.5.13 使用分类处理其他格式 221 14.5.14 在iOS中提升性能的小技巧 222 14.6 小结 222 14.7 扩展阅读 222 14.7.1 苹果文档 222 14.7.2 其他资源 223 第15章 通过安全服务巩固系统安全 224 15.1 理解iOS沙盒 224 15.2 保证网络通信的安全 225 15.2.1 证书工作原理 226 15.2.2 检验证书的有效性 228 15.2.3 判断证书的可信度 231 15.3 使用文件保护 233 15.4 使用钥匙串 234 15.5 使用加密 236 15.5.1 AES概要 237 15.5.2 使用PBKDF2将密码转换成密钥 237 15.5.3 AES模式和填充 239 15.5.4 初始化向量 240 15.5.5 使用HMAC进行认证 241 15.5.6 错误的密码 241 15.5.7 进行单槽加密 242 15.5.8 提高CommonCrypto的性能 244 15.5.9 组合使用加密和压缩 247 15.6 小结 247 15.7 扩展阅读 248 15.7.1 苹果文档 248 15.7.2 WWDC讲座 248 15.7.3 其他资源 248 第16章 在多个苹果平台和苹果设备上运行应用 249 16.1 开发多平台应用 249 16.1.1 可配置的目标设置：BaseSDK和DeploymentTarget 250 16.1.2 支持多个SDK时的注意事项：框架、类和方法 250 16.1.3 检查框架、类和方法的可用性 252 16.2 检测设备的功能 253 16.2.1 检测设备及判断功能 253 16.2.2 检测硬件和传感器 254 16.3 应用内发送Email和短信 258 16.4 检查多任务支持 258 16.5 支持iPhone5 259 16.6 UIRequiredDeviceCapabilities 260 16.7 小结 261 16.8 扩展阅读 262 16.8.1 苹果文档 262 16.8.2 其他资源 262 第17章 国际化和本地化 263 17.1 什么是本地化 263 17.2 本地化字符串 264 17.3 对未本地化的字符串进行审查 265 17.4 格式化数字和日期 266 17.5 nib文件和BaseInternationalization 269 17.6 本地化复杂字符串 270 17.7 小结 272 17.8 扩展阅读 272 17.8.1 苹果文档 272 17.8.2 WWDC讲座 272 第18章 通过应用内购买提高销售业绩 273 18.1 开始之前 273 18.2 应用内购买什么 273 18.2.1 禁止通过应用内购买出售的产品 274 18.2.2 重新思考你的商业模式 275 18.3 在iTunesConnect上设置产品配置 275 18.3.1 第一步：为你的应用创建一个新的AppID 276 18.3.2 第二步：生成配置文件 277 18.3.3 第三步：创建应用的产品项 277 18.3.4 第四步：创建应用内购买产品项 278 18.3.5 第五步：生成共享密钥 279 18.3.6 第六步：创建测试账户 279 18.3.7 第七步：创建托管内容 279 18.4 实现应用内购买 281 18.5 MKStoreKit简介 281 18.5.1 为什么要用MKStoreKit 281 18.5.2 MKStoreKit的设计 282 18.5.3 定制MKStoreKit 283 18.6 实现购买 285 18.7 下载托管内容 285 18.8 测试代码 286 18.9 排错 286 18.9.1 无效的产品ID 286 18.9.2 无法连接iTunesStore 286 18.9.3 你已经购买了该产品，但它尚未下载 287 18.10 小结 287 18.11 扩展阅读 287 18.11.1 苹果文档 287 18.11.2 博客 287 18.11.3 其他资源 287 第19章 调试 288 19.1 LLDB 288 19.2 使用LLDB进行调试 288 19.2.1 dSYM文件 289 19.2.2 符号化 290 19.3 断点 291 19.4 观察点 294 19.5 LLDB控制台 295 19.6 NSZombieEnabled标志 298 19.7 不同的崩溃类型 299 19.7.1 EXC_BAD_ACCESS 299 19.7.2 SIGSEGV 300 19.7.3 SIGBUS 300 19.7.4

SIGTRAP 300 19.7.5 EXC_ARITHMETIC 300 19.7.6 SIGILL 300 19.7.7 SIGABRT 301 19.7.8 看门狗超时 301 19.7.9 自定义错误信号处理程序 301 19.8 收集崩溃报告 302 19.9 第三方崩溃报告服务 303 19.10 小结 303 19.11 扩展阅读 304 19.11.1 苹果文档 304 19.11.2 WWDC讲座 304 19.11.3 其他资源 304 第20章 性能调优 305 20.1 性能思维模式 305 20.1.1 指导方针一：产品是为了取悦用户才存在的 305 20.1.2 指导方针二：设备是为了方便用户而存在的 305 20.1.3 指导方针三：做到极致 305 20.1.4 指导方针四：用户的感知才是实际的 306 20.1.5 指导方针五：关注能带来大收益的方面 306 20.2 欢迎走入Instruments的世界 306 20.3 查找内存问题 308 20.4 查找CPU问题 311 20.4.1 Accelerate框架 314 20.4.2 GLKit 315 20.5 绘图性能 315 20.6 优化磁盘访问和网络访问 317 20.7 小结 318 20.8 扩展阅读 318 20.8.1 苹果文档 318 20.8.2 其他资源 318 第四部分 超越极限 第21章 故事板及自定义切换效果 320 21.1 初识故事板 320 21.1.1 实例化故事板 321 21.1.2 加载故事板中的视图控制器 321 21.1.3 联线 321 21.1.4 使用故事板来实现表视图 323 21.2 自定义切换效果 324 21.2.1 还有一个优点 325 21.2.2 白璧微瑕 326 21.3 使用UIAppearance协议自定义视图 326 21.4 小结 327 21.5 扩展阅读 327 21.5.1 苹果文档 327 21.5.2 WWDC讲座 327 21.5.3 其他资源 327 第22章 Cocoa的大招：键值编码和观察 328 22.1 键值编码 328 22.1.1 用KVC赋值 330 22.1.2 用键路径遍历属性 331 22.1.3 KVC和容器类 331 22.1.4 KVC和字典 335 22.1.5 KVC和非对象 336 22.1.6 用KVC实现高阶消息传递 336 22.1.7 容器操作符 336 22.2 键值观察 337 22.2.1 KVO和容器类 339 22.2.2 KVO是如何实现的 340 22.3 KVO的权衡 340 22.4 小结 342 22.5 扩展阅读 342 22.5.1 苹果文档 342 22.5.2 其他资源 342 第23章 不同凡想：块和函数式编程 343 23.1 什么是块 343 23.1.1 为什么要用函数式编程 343 23.1.2 “ 函数式 ” UIAlertView 344 23.2 声明块 345 23.2.1 变量作用域 346 23.2.2 栈与堆 346 23.3 实现块 347 23.4 块和并发 349 23.4.1 GCD中的分派队列 349 23.4.2 NSOperationQueue与GCD分派队列 350 23.5 基于块的Cocoa方法 351 23.5.1 UIView的动画使用块 351 23.5.2 展示和移除视图控制器 352 23.5.3 TweetComposer与应用程序中发送邮件（短信） 352 23.5.4 用NSDictionary的enumerationWithBlock遍历字典 352 23.5.5 寻找基于块的方法 353 23.6 支持情况 353 23.7 小结 353 23.8 扩展阅读 354 23.8.1 苹果文档 354 23.8.2 博客 354 23.8.3 其他资源 354 第24章 离线支持 355 24.1 需要离线支持的原因 355 24.2 缓存策略 356 24.2.1 存储缓存 356 24.2.2 缓存版本和失效 359 24.3 数据模型缓存 359 24.4 缓存版本控制 363 24.5 创建内存缓存 364 24.5.1 为AppCache设计内存缓存 365 24.5.2 处理内存警告 366 24.5.3 处理结束和进入后台通知 367 24.6 创建URL缓存 367 24.6.1 过期模型 368 24.6.2 验证模型 368 24.6.3 示例 368 24.6.4 用URL缓存来缓存图片 369 24.7 小结 369 24.8 扩展阅读 369 24.8.1 苹果文档 369 24.8.2 书籍 369 24.8.3 其他资源 369 第25章 云端数据 370 25.1 iCloud 370 25.2 第三方云服务提供商 372 25.3 Parse 373 25.3.1 Parse入门 373 25.3.2 Parse的顶层对象 374 25.3.3 代码 374 25.4 StackMob 376 25.4.1 设置StackMob 376 25.4.2 登录、上传和获取数据 377 25.4.3 StackMob自定义代码 377 25.5 后端即服务的缺点 378 25.6 小结 378 25.7 扩展阅读 378 25.7.1 苹果文档 378 25.7.2 WWDC讲座 378 25.7.3 其他资源 379 第26章 精妙的文本布局 380 26.1 基本控件：字段、视图和标签 380 26.2 UIKit中的富文本 381 26.2.1 理解粗体、斜体和下划线 381 26.2.2 属性化字符串 382 26.2.3 段落样式 384 26.2.4 属性化字符串和HTML 384 26.3 用Web视图显示富文本 385 26.3.1 在Web视图中显示和访问HTML 385 26.3.2 响应用户交互 386 26.3.3 在滚动视图和表格视图中绘制Web视图 386 26.4 CoreText 387 26.4.1 用CTFramesetter进行简单的布局 387 26.4.2 为非连续路径创建框架 388 26.4.3 排版器、文本行、连续文本和字形 390 26.4.4 沿着曲线绘制文本 391 26.5 小结 395 26.6 扩展阅读 395 26.6.1 苹果文档 395 26.6.2 WWDC讲座 395 26.6.3 其他资源 396 第27章 创建CoreFoundation应用 397 27.1 CoreFoundation类型 397 27.2 命名和内存管理 398 27.3 分配器 399 27.4 内省 400 27.5 字符串和数据 400 27.5.1 常量字符串 401 27.5.2 创建字符串 401 27.5.3 转换为C字符串 402 27.5.4 其他字符串操作符 404 27.5.5 字符串的支持存储 404 27.5.6 CFData 405 27.6 容器类型 405 27.6.1 CFArray 406 27.6.2 CFDictionary 406 27.6.3 CFSet和CFBag 406 27.6.4 其他容器类型 407 27.6.5 回调函数 407 27.7 自由桥接 408 27.8 小结 411 27.9 扩展阅读 411 27.9.1 苹果文档 411 27.9.2 其他资源 411 第28章 深度解析Objective—C 412 28.1 理解类和对象 412 28.2 使用方法和属性 414 28.3 消息传递如何工作 416 28.3.1 动态实现 417 28.3.2 快速转发 419 28.3.3 普通转发 422 28.3.4 转发失败 422 28.3.5 各种版本的objc_msgSend 423 28.4 方法混写 423 28.5 ISA混写 426 28.6 方法混写与ISA混写 427 28.7 小结 427 28.8 扩展阅读 427 28.8.1 苹果文档 427 28.8.2 其他资源 428 索引 429

章节摘录

版权页： 插图： 在iOS应用开发中使用Git，请参考以下步骤：用主分支来映射与App Store中版本对应的最新最好的代码；对于每一个新版本，创建一个新的版本分支；对于每一个主要功能，在相应的版本分支上再创建一个功能分支；功能开发完成时将功能分支合并到相应的版本分支；把应用提交到App Store时，将相应的版本分支合并到主分支；当应用审核通过时，给主分支打一个标签，这一步是可选的。
完成这些步骤总共只需要几分钟而已。
按照这个步骤，你就可以非常容易地在某个版本分支上进行bug修复并且将代码变更合并到当前工作分支上。
举个例子，假如苹果拒绝了你的应用，你需要做的就是签出主分支，修复问题，重新提交到AppStore，然后签出当前工作分支并将代码变更合并进去。
使用Git，非线性软件开发过程会变得非常容易。
试用一下Git，你不会后悔的。
Xcode 4中最强大也最让人困惑的新功能就是Scheme。
在Xcode 3中，应用运行前可以在编译配置下拉列表中指定当前编译配置、当前目标、当前可执行文件、当前架构（指令集），以及目标设备。
即使是默认的那组选项也会产生大量的选择组合。
而对于复杂的工程设置，针对目标平台（或设备）和指令集选用正确的可执行文件就会很困难。
最糟糕的是，Xcode 3允许为目标选择错误的可执行文件。
Xcode 4引入Scheme来帮助开发者轻松处理这些问题。
一个Scheme就是前面提过的各种设置的一个集合体，一个Scheme就是编译一个产品的指令集合。
这个产品（大多数时候都）可以是目标和其编译配置的一个集合体。
你也可以为这些目标指定已存在的xcconfig文件。
本章后面会讲述xcconfig配置文件。
之前，如果要编译一些东西，需要选择4个不同的选项，保证每次都正确是非常困难的。
有些时候，你可能会将调试版本的应用提交到App Store，也可能拿发布版本的应用做调试，然后发现断点没有生效。
现在有了Scheme，你只需要选择一个Scheme就可以了，其他的选项都会根据相应的Scheme自动设置好。
当准备把编译的产品用于调试时，很显然，你不希望调试语句被删除。
相反，当为AppStore编译产品时，通常你会做一些性能优化并且删除调试语句。
这点对于自发布（非App Store方式）部署也是非常好的。
等等！
Scheme能做的远不止这些，你还可以把Scheme提交到版本库中，这样就可以将官们在同事之间共享了。

媒体关注与评论

《iOS 6编程实战》不是一本简单的基础教程或入门书籍，它教会读者用正确的方法来解决每个独特的问题，两位作者结合自己的实际开发经验将这些知识整理、归纳并清楚地呈现在读者面前。
《iOS 6编程实战》从安全、多任务处理、在多平台上运行、模块和函数式编程、高级文本布局等多个方面讲述了如何为iPad、iPhone和iPod touch打造杀手级应用、如何将性能最大化并从中赚取最大的价值。

编辑推荐

《图灵程序设计丛书:iOS 6编程实战》编辑推荐：深度揭秘iOS 6高级特性与开发技巧；发掘最佳实践，借鉴开发经验，真正掌控iOS开发；移动开发进阶必备，在最新平台上打造非凡应用。
久经“杀场”的两位作者Rob Napier和Mugunth Kumar在《图灵程序设计丛书:iOS 6编程实战》中分享了他们为苹果移动设备开发“杀手级”应用的宝贵经验。

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:http://www.tushu007.com