<<ios开发实战体验>>

图书基本信息

书名：<<ios开发实战体验>>

13位ISBN编号：9787502783112

10位ISBN编号：7502783113

出版时间：2012-8

出版时间：海洋出版社

作者：DevDiv移动开发社区

页数：324

前言

苹果iOS系统已经历5年多17个版本的更新，其在多代iPhone、iPad和iPod设备上取得巨大成功，目前已经成长为市场上影响力最大、功能最丰富、生态最完整的移动操作系统。

iOS的成功得益于苹果公司对移动互联网的深刻理解，苹果公司始终专注于用户体验与服务，并通过平台整合构建了完整的应用生态链。

iOS从诞生之初即专注用户体验与服务的提升，通过不断的技术革新，一次次引领着UI交互方式的变革。

比如，Siri技术就是 iOS5 最大的亮点，实现了语音控制输入功能，可以通过Siri使用语音提问和评论，并且可以与包括日历等在内的所有应用通信。

同时，新“retina”显示屏像素要比历史版本的显示屏像素高出3倍左右，将用户体验提升到前所未有的高度，同时，也拉大了与竞争对手之间的距离！

APP Store是苹果公司构建应用生态链的关键形式，它让众多的开发者找到了自己的商业模式和商业机会，而苹果公司因此积累了庞大数量的应用。

云服务iCloud让所有的iOS设备实现了互联互通，通过iCloud，使用同一账号的用户可以在不同iOS设备上同步信息和APP，实现文件备份、存储等功能。

可以看出，以iCloud为“媒”，统一不同设备系统平台是苹果公司的既定战略，苹果公司在平台整合上已经迈出了坚实的一步。

在iOS过去 5 年多时间里，业界对苹果公司的疑问从“这是智能手机吗”发展到了“它可以代替PC”吗，可见其对智能手机发展的贡献，现在已经没有人怀疑以iOS引领的新一代智能手机将成为最重要的个人计算终端、个人娱乐终端和个人通信终端，其地位将与传统 PC 分庭抗礼。

苹果公司依靠其强大的垂直一体化战略，不仅通过iPhone、iPad等产品赚取了高额的利润，同时，通过吸引开发者不断地提供创新的应用，为苹果公司源源不断地注入活力。

此外，值得关注的是，苹果公司并不单纯是“以质取胜”，在数量上，苹果公司也有望在未来几年实现对Windows 设备（安装微软Windows操作系统的设备）的反超。

相关统计数据显示：过去，Windows设备在同苹果设备的销量对比上，一直占据绝对优势，这一优势在2000年左右曾经达到过一个峰值，每销售50台Windows设备才售出一款苹果设备。

但是，随着iPad和iPhone等一系列基于移动互联产品的问世，这一数字比例在急剧减小，截止目前，Windows设备与苹果设备的销量比例约为2：1，分析师称，苹果设备数量有望在未来两年内超过微软。

当前，基于iOS的应用具有广阔的前景和良好的发展趋势。

希望广大开发者借iOS之势，利用《iOS开发实战体验》及书中提供的相关代码，快速进入iOS开发领域，实现开发者梦想！

中国科学院博士、博士生导师，软件研究所研究员 金蓓弘

<<ios开发实战体验>>

内容概要

iOS是移动开发三大平台之一。
本书是DevDiv移动开发社区版主、资深会员继成功推出《移动开发全平台解决方案——Android/iOS/Windows Phone》和《Windows Phone开发实战体验（应用+游戏）》后的又一力作。
该书章节设置全面涵盖iOS应用开发技术要点，原创案例细致呈现作者实际开发经验。
具体包括iOS开发框架、Object-C高级知识、MVC设计和Push消息、视图高级使用技巧、数据持久化、TableView使用、文件I/O、硬件和通信、iOS多媒体、定位和地图、网络编程、连接到互联网、多线程编程、2D和3D绘图编程、调试和优化等内容。

作者简介

DevDiv覆盖移动开发主流平台Android、iOS、Windows Phone、Windows 8、HTML5和Symbian等，集资讯、论坛、博客、猎头服务、技术培训于一体，是国内最具人气的综合性移动开发社区。

<<ios开发实战体验>>

书籍目录

章节摘录

版权页： 插图： 业务模型还有一个很重要的模型，那就是数据模型。
数据模型主要指实体对象的数据保存（持续化）。
比如将一张订单保存到数据库，需要时从数据库获取订单。
我们可以将这个模型单独列出，所有有关数据库的操作只限制在该模型中。
（2）视图（View）层 视图即为用户交互界面。
对于Web应用来说，可以是HTML界面，也有可能是XHTML、XML和Applet界面。
随着应用复杂程度和规模的提升，界面的处理也变得越来越有挑战性。
一个应用可能有很多不同的视图，MVC仅限于视图上数据的采集、处理以及用户的请求，而不包括视图上业务流程的处理。
业务流程交由模型（Model）处理。
比如一个订单的视图只接受来自模型的数据并进行显示，以及将用户界面的输入数据和请求传递给控制器和模型。
（3）控制器（Controller）层 控制器主要负责从用户接收请求，并将模型与视图整合在一起，共同完成相关任务。
它像一个分发器，清楚地告诉开发者，可以选择什么样的模型和什么样的视图，完成什么样的用户请求。
控制层本身不做任何的数据处理，例如用户点击一个链接，控制层接受请求后，并不处理业务信息，只是把用户的信息传递给模型，告诉模型做什么，然后选择符合要求的视图返回给用户。
因此，一个模型可能对应多个视图，一个视图也可能对应多个模型。
2）MVC的优点 MVC要求对应用分层，虽然会增加额外的工作，但产品的结构清晰，其应用通过模型可以得到更好的体现。
具体表现如下：①具有多个视图对应一个模型的能力。
在目前用户需求快速变化的情况下，可能希望通过多种方式访问应用。
例如，订单模型可能有本系统的订单，也有网上订单，或者其他系统的订单，不管是哪种，对订单的处理都是一样的，也就是说订单的处理是一致的。
按MVC设计模式，一个订单模型及多个视图即可解决问题。
这样既减少了代码的重复，又减少了代码的维护量，一旦模型发生改变，也易于维护。
②由于模型返回的数据不带任何显示格式，为此这些模型也可直接应用于接口。
③由于一个应用被分离为三层，因此有时仅改变其中的一层就能满足应用的变化需求。
当应用的业务流程或者业务规则改变时只需改动MVC的模型层，其他两层可保持不变。
④由于控制层是把不同的模型和不同的视图组合在一起完成不同的请求，因此，控制层可以说是包含了用户请求权限的概念。
⑤MVC模式有利于软件工程化管理。
由于不同的层各司其职，每一层的不同应用间具有某些相同的特征，有利于通过工程化、工具化产生管理程序代码。
3）MVC的不足 MVC的不足体现在以下几个方面：①增加了系统结构和实现的复杂性。
对于简单的界面，如果严格遵循MVC模式，使模型、视图与控制器分离，会增加结构的复杂性，并可能产生过多的更新操作，降低运行效率。
②视图与控制器的联系仍过于紧密。
视图与控制器是既相互分离又确实联系紧密的部件，没有控制器，视图的应用会很有限，反之亦然，这样就妨碍了它们的独立重用。
③视图对模型数据的访问效率比较低。
依据模型操作接口的不同，视图可能需要多次调用才能获得足够的显示数据。
对未变化数据的不必要的频繁访问，也降低了操作性能。

编辑推荐

《移动开发技术丛书:iOS开发实战体验》适用于iOS初中级开发者参考用书、高等院校及社会培训机构教材、自学人员学习用书。

<<ios开发实战体验>>

版权说明