

<<20天搞定iPhone软件开发>>

图书基本信息

书名：<<20天搞定iPhone软件开发>>

13位ISBN编号：9787121184871

10位ISBN编号：7121184877

出版时间：2012-10

出版时间：电子工业出版社

作者：王志刚 主编

页数：387

字数：510000

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>

<<20天搞定iPhone软件开发>>

内容概要

本书是为有一定其他语言编程经验的程序员转向iOS软件开发而撰写的速成教材。全书一共20讲，一天完成1讲的学习。本着循序渐进的原则，前面的10讲偏重于Cocoa面向对象编程的基本概念以及Objective-C 2.0语言的语法基础。

从第10讲以后开始进入iOS软件开发的实战阶段，每1讲的内容都比较充实，其中包含了各种实例代码，读者可以在Xcode等开发环境中边实践边学习。读者完成全书20讲的学习后可以达到初级iOS程序员的水准。未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。

<<20天搞定iPhone软件开发>>

书籍目录

第1讲 配置开发环境 1

- 1.1 Mac应用程序开发环境 1
- 1.2 iPhone应用程序开发环境 2
- 1.3 查看安装目录内容 .4
 - 1.3.1启动 Xcode 5
 - 1.3.2启动 Interface Builder 6
- 1.4 Mac中的基本操作 6
 - 1.4.1弹出菜单的显示方法 6
 - 1.4.2工具条的定制

第2讲 应用程序与框架 .10

- 2.1 关于应用程序 .10
 - 2.1.1应用程序的种类 .11
 - 2.1.2关于 iPhone应用程序 . 12
- 2.2 框架 .13
 - 2.2.1简单的【 Hello World】程序 . 13
 - 2.2.2框架的出现 . 13
 - 2.2.3编程语言与框架的关系 14
- 2.3 MacOS X以及 iPhone SDK中的框架 14
 - 2.3.1框架的安装目录 14
 - 2.3.2 Cocoa与Carbon . 15
 - 2.3.3应用程序中使用的框架 16
- 2.4 Cocoa与Carbon的发展轨迹

第3讲 Cocoa与Objective-C20

- 3.1 框架的使用 .20
 - 3.1.1关于 API 20
 - 3.1.2 API与编程语言的关系
- 3.2 Objective-C的发展史22
- 3.3 用于其他语言中的Cocoa API 23
 - 3.3.1通过桥调用Cocoa . 23
 - 3.3.2是否应提倡在Objective-C以外的语言中调用Cocoa API的开发方式 . 24
- 3.4 解答与Objective-C相关的几个疑问

第4讲 开发工具27

- 4.1 Xcode .27
 - 4.1.1综合开发环境Xcode的功能 . 27
 - 4.1.2工程窗口. 28
 - 4.1.3文本编辑器的功能 32
 - 4.1.4开发者手册 . 35
- 4.2 Interface Builder 36
- 4.3 Dashcode37
- 4.4 iPhone模拟器 38
- 4.5 其他工具 .38
 - 4.5.1 FileMerge 38
 - 4.5.2 Icon Composer 39
 - 4.5.3 Property List Editor. 39
- 4.6 Mac或iPhone应用程序开发的流程 .39

<<20天搞定iPhone软件开发>>

- 4.6.1准备应用程序用的材料 39
- 4.6.2建立步骤. 41
- 4.6.3 gcc
- 第5讲 Cocoa环境下的面向对象编程.43
- 5.1 面向对象的基本概念术语 .43
- 5.1.1类 44
- 5.1.2实例 44
- 5.1.3实例变量. 44
- 5.1.4方法 45
- 5.1.5继承 45
- 5.1.6相互关系. 46
- 5.1.7对象 46
- 5.2 在Cocoa实例中理解基本术语 .47
- 5.2.1通过按钮配置理解基本术语 . 47
- 5.2.2文本输入框的继承关系
- 第6讲 Hello World应用程序49
- 6.1 Hello World程序
- 6.2 启动Xcode创建工程 .49
- 6.3 编辑代码 .51
- 6.4 启动Interface Builder设计用户界面 52
- 6.4.1 xib文件 . 52
- 6.4.2 Interface Builder窗口 53
- 6.4.3设计Hello World的用户界面. 55
- 6.5 连接插座与动作 .55
- 6.6 完成Hello World App Delegate的代码编写58
- 6.7 建立与运行 .59
- 6.8 开发步骤再回顾
- 第7讲 MVC构架.62
- 7.1 HelloWorld应用程序开发流程中包含的信息.62
- 7.2 什么是MVC构架 62
- 7.2.1视图层 63
- 7.2.2模型层 63
- 7.2.3控制层 64
- 7.3 使用MVC构架的实例 64
- 7.3.1 iTunes的模型 64
- 7.3.2 iTunes的视图 65
- 7.3.3 iTunes的控制 66
- 7.4 使用MVC构架的Cocoa应用程序开发特点66
- 7.4.1视图组件由Cocoa提供 66
- 7.4.2保持视图与模型间的相互独立 . 67
- 7.4.3应用程序特有的功能由控制提供 . 67
- 7.4.4只用创建模型以及控制的类
- 第8讲 插座与动作68
- 8.1 视图与控制间信息交换 .68
- 8.2 插座与动作 .69
- 8.2.1什么是插座 . 69
- 8.2.2什么是动作 . 69

<<20天搞定iPhone软件开发>>

- 8.3 插座以及动作的连接 70
 - 8.3.1插座与动作的追加 70
 - 8.3.2插座的连接 . 71
 - 8.3.3动作的连接 . 72
 - 8.3.4连接确认.
- 第9讲 Objective-C的语法
 - 9.1 Objective-C的编程概论
 - 9.1.1 Objective-C与C语言
 - 9.1.2 Objective-C程序
 - 9.1.3 类的声明与实体
 - 9.1.4 创建Objective-C的测试工程
 - 9.2 类的声明
 - 9.2.1 导入声明文件
 - 9.2.2 类的声明
 - 9.2.3 实例变量的声明
 - 9.2.4 实例方法的声明
 - 9.3 编写类的实体
 - 9.3.1 编写类的实体
 - 9.3.2 编写方法代码
 - 9.3.3 关于方法的声明
 - 9.4 对象专用变量类型
 - 9.4.1 实例对象专用的变量类型
 - 9.4.2 id类型
 - 9.4.3 nil类型
 - 9.4.4 在条件表达式中使用对象变量
 - 9.5 方法调用
 - 9.5.1 实例方法的调用
 - 9.5.2 类方法的调用
 - 9.5.3 对象为nil的情况
 - 9.6 命名规则
 - 9.6.1 类的命名规则
 - 9.6.2 实例变量的命名规则
 - 9.6.3 方法的命名规则
 - 9.7 重新解析Cocoa的Hello World程序
 - 9.7.1 HelloWorldAppDelegate.h
 - 9.7.2 HelloWorldAppDelegate.m
- 第10讲 内存管理
 - 10.1 内存分配与释放
 - 10.2 与内存相关的问题
 - 10.2.1 内存释放的时机
 - 10.2.2 内存泄漏
 - 10.3 使用垃圾收集进行内存管理
 - 10.3.1 垃圾回收
 - 10.3.2 Objective-C与垃圾回收
 - 10.3.3 垃圾回收功能有效化
 - 10.3.4 垃圾回收编程的规则
 - 10.4 由参照统计实现的内存管理

<<20天搞定iPhone软件开发>>

- 10.4.1 参照统计的基本概念
- 10.4.2 retain、release方法
- 10.4.3 对象保持到释放的流程
- 10.5 类的实例化
 - 10.5.1 类的实例化与初期化
 - 10.5.2 自动释放
 - 10.5.3 创建实例的方法
 - 10.5.4 实例的释放
- 10.6 内存管理的规则
 - 10.6.1 临时对象
 - 10.6.2 内存管理经验总结
- 第11讲 字符串
 - 11.1 字符串专用类
 - 11.2 NSString与NSMutableString
 - 11.3 字符编码
 - 11.3.1 什么是字符编码
 - 11.3.2 NSString的字符代码为Unicode
 - 11.3.3 NSStringEncoding
 - 11.4 字符串的生成
 - 11.4.1 使用【@ ” ”】的形式创建新字符串
 - 11.4.2 创建中文字符串
 - 11.4.3 使用格式创建字符串
 - 11.5 NSRange
 - 11.6 字符串处理
 - 11.6.1 计算字符串的长度
 - 11.6.2 字符串连接、插入、删除
 - 11.6.3 字符串的比较
 - 11.6.4 字符串的检索
 - 11.6.5 抽取部分字符串
 - 11.7 读写文本文件
 - 11.7.1 读取文本文件
 - 11.7.2 文本文件输出
- 第12讲 集合
 - 12.1 数组
 - 12.1.1 NSArray与 NSMutableArray
 - 12.1.2 数组的创建
 - 12.1.3 数组的长度
 - 12.1.4 通过索引取得对象
 - 12.1.5 使用NSEnumerator取得对象
 - 12.1.6 使用高速枚举取得对象
 - 12.1.7 对象的追加与删除
 - 12.2 字典
 - 12.2.1 什么是字典类型
 - 12.2.2 NSDictionary与 NSMutableDictionary
 - 12.2.3 NSDictionary的创建
 - 12.2.4 键与值的取得
 - 12.2.5 键与值的追加

12.3 包装类

12.3.1 什么是包装类

12.3.2 NSNumber

12.4 对象的等价与同值

12.4.1 拥有相同值的对象

12.4.2 等价与

章节摘录

版权页：插图：使用这个垃圾决定规则后，垃圾回收过程就变成如下的方式了。

首先，启动应用程序。

在一定的时间内，按照自己的方式创建对象，将创建完成的对象放在一边。

当创建的对象的数量积累到一定的程度，暂时停止应用程序的运行，启动垃圾回收程序。

垃圾回收程序从所有的根对象开始追踪，在需要的对象上做标记。

最后将没有标记的对象当做垃圾，一次性回收。

增加了应用程序可使用的内存空间后，再重新开始运行应用程序。

10.3.2 Objective—C与垃圾回收 像这样如此方便的垃圾回收功能，在Cocoa+Objective—C环境中使用了吗？

答案是肯定的。

但是，是从Objective—C 2.0开始的。

也就是说，Mac OS X 10.5以前的版本中是没有使用的。

因此，从现在开始，如果要开发新的应用程序则可以使用垃圾回收的功能，而且也建议使用。

因为这个功能真的很方便。

另外，也存在不应使用垃圾处理的情况。

1.如果需要在Mac OS X 10.4中运行时 如果应用程序要运行在Mac OS X 10.4版本以及更旧的版本中，则垃圾回收处理将不能使用，因为垃圾回收必须在Objective—C 2.0中使用。

2.在过去代码基础上开发的情况下 如果不是新开发的应用程序，而是在过去的应用程序的基础上开发的情况下，则最好放弃使用垃圾回收的功能。

因为，将使用垃圾回收的代码与不使用垃圾回收的代码混在一起，管理起来非常麻烦。

有人会说，将旧代码部分也改造成使用垃圾回收的形式不就可以了吗？

最好别那么做，辛辛苦苦调试好的代码，最好还是原封不动直接使用为好。

另外，如果将其改造成使用垃圾回收的形式，还可能会出现后面将会介绍的性能问题，所以还是不要蛮干。

3.以性能作为最重要指标的应用程序 垃圾回收是很方便的，但是有一个缺点就是影响性能。

垃圾回收的机制是在进行垃圾回收的时候，必须停止应用程序的运行。

而且这个垃圾回收的过程还是非常费时间的，因为你必须检查已存在的几万个，甚至几十万个对象。

如果应用程序非常重视性能，有时会忍痛放弃使用垃圾回收处理，而采取手动的内存管理方式。

像这样的应用程序有需要实时处理的系统，以及游戏这样需要立即反应的应用程序等。

另外，像嵌入式设备（iPhone就是嵌入式设备），其本身CPU的处理速度就慢，内存也没有多少，最好不要使用垃圾回收功能。

<<20天搞定iPhone软件开发>>

编辑推荐

《20天搞定iPhone软件开发(适用于iOS 5.0)(双色版)》编辑推荐：iPhone开发基础入门必备图书。

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>