

<<Objective-C编程之道>>

图书基本信息

书名：<<Objective-C编程之道>>

13位ISBN编号：9787115265869

10位ISBN编号：7115265860

出版时间：2011-11

出版时间：人民邮电

作者：钟冠贤

译者：刘威

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>

<<Objective-C编程之道>>

内容概要

本书是基于iOS 的软件开发指南。
书中应用GoF
的经典设计模式，介绍了如何在代码中应用创建型模式、结构型模式和行为模式，如何设计模式以巩固应用程序，并通过设计模式实例介绍MVC
在CocoaTouch 框架中的工作方式。

本书适用于那些已经具备Objective-C 基础、想利用设计模式来提高软件开发效率的中高级iOS 开发人员。

苹果公司的App
Store拥有超过20万个应用（每秒都在增加）供用户选择，正深刻地改变着软件行业。
每天都有更多的iOS开发者想投入到这一潮流，希望凭藉下一个杀手级应用发家致富。
本书的目标正是带领读者完成从新手到高手的转变，关注底层的设计模式而非一味只顾着写代码，从而开发出更加高效、实用和专业的应用。

iOS 应用程序的基础Cocoa
Touch框架内容丰富、结构优美，通过将各种设计模式应用到其基础结构中，为第三方开发者提供了很好的可扩展性和灵活性。
因此，要充分利用这一框架，应当深刻理解并恰当应用设计模式。
本书受到GoF的经典著作《设计模式》的启发，旨在引导大家掌握如何在iOS平台上以Objective-C语言实现
Cocoa Touch开发所要用到的传统设计模式。

在编写代码的过程中，你可能在一定程度上运用了一些设计模式，只是并没有意识到或充分利用它们。
基于此，本书深入解析了这些设计模式。
特定模式方法的实现将向iOS应用开发人员展示其非凡价值。
你将掌握单例、抽象工厂、责任链和观察者等经典模式，还会发现一些不太知名但非常有用的模式，比如备忘录、组合、命令和中介者等。

学完本书，你将学会：

- ?各种设计模式的基本概念；
- ?根据不同场景，将设计模式应用于代码中；
- ?用设计模式来改进应用程序；
- ?提高软件开发的效率。

<<Objective-C编程之道>>

作者简介

Carlo

Chung (钟冠贤) 计算机科学家, 喜欢摆弄机器、爱好摄影。

他拥有计算机科学硕士学位, 专业方向是计算机视觉(人工智能的分支)。

把人工智能与任何小装置结合起来的想法都能令他兴奋不已。

他常常梦想着让iPhone变得更智能、更像人, 曾把计算机视觉的知识应用到iPhone平台并开发了几个应用, 有两个被作为特色应用出现在App

Store首页, 其中有一个还被列为摄影类别中的最佳付费应用(Top Paid)。

译者简介:

刘威

2001年毕业于中国科学院研究生院。

热爱计算机科学, 喜欢编程, 尤其喜欢写程序解决实际问题。

从2000年起从事软件开发工作, 最近几年一直专注于手机软件开发, 目前从事Android和iOS软件开发。

新浪微博: @刘威-LiuWei。

<<Objective-C编程之道>>

书籍目录

第一部分 设计模式初体验

第1章 你好，设计模式

- 1.1 这是一本什么书
- 1.2 开始前的准备
- 1.3 预备知识
- 1.4 似曾相识的设计
- 1.5 设计模式的起源——模型、视图和控制器
 - 1.5.1 在模型对象中封装数据和基本行为
 - 1.5.2 使用视图对象向用户展示信息
 - 1.5.3 用控制器对象联系起模型和视图
 - 1.5.4 作为复合设计模式的MVC
- 1.6 影响设计的几个问题
 - 1.6.1 针对接口编程，而不是针对实现编程
 - 1.6.2 @protocol 与抽象基类
 - 1.6.3 对象组合与类继承
- 1.7 本书用到的对象和类
 - 1.7.1 类图
 - 1.7.2 对象图
- 1.8 本书如何安排模式的讲解
- 1.9 总结

第2章 案例分析：设计一个应用程序

- 2.1 想法的概念化
- 2.2 界面外观的设计
- 2.3 架构设计
 - 2.3.1 视图管理
 - 2.3.2 如何表现涂鸦
 - 2.3.3 如何表现保存的涂鸦图
 - 2.3.4 用户操作
- 2.4 所用设计模式的回顾
- 2.5 总结

第二部分 对象创建第3章 原型

- 3.1 何为原型模式
- 3.2 何时使用原型模式
- 3.3 浅复制与深复制
- 3.4 使用Cocoa Touch 框架中的对象复制
- 3.5 为Mark 聚合体实现复制方法
- 3.6 将复制的Mark 用作“图样模板”
- 3.7 总结

第4章 工厂方法

- 4.1 何为工厂方法模式
- 4.2 何时使用工厂方法
- 4.3 为何这是创建对象的安全方法
- 4.4 在TouchPainter 中生成不同画布
- 4.5 在Cocoa Touch 框架中应用工厂方法
- 4.6 总结

<<Objective-C编程之道>>

第5章 抽象工厂

- 5.1 把抽象工厂应用到TouchPainter应用程序
- 5.2 在Cocoa Touch 框架中使用抽象工厂
- 5.3 总结

第6章 生成器

- 6.1 何为生成器模式
- 6.2 何时使用生成器模式
- 6.3 构建追逐游戏中的角色
- 6.4 总结

第7章 单例

- 7.1 何为单例模式
- 7.2 何时使用单例模式
- 7.3 在Objective-C 中实现单例模式
- 7.4 子类化Singleton
- 7.5 线程安全
- 7.6 在Cocoa Touch 框架中使用单例模式
 - 7.6.1 使用UIApplication 类
 - 7.6.2 使用UIAccelerometer 类
 - 7.6.3 使用NSFileManager 类
- 7.7 总结

第三部分 接口适配

第8章 适配器

- 8.1 何为适配器模式
- 8.2 何时使用适配器模式
- 8.3 委托
- 8.4 用Objective-C 协议实现适配器模式
- 8.5 用Objective-C 的块在iOS 4 中实现适配器模式
 - 8.5.1 块引用的声明
 - 8.5.2 块的创建
 - 8.5.3 把块用作适配器
- 8.6 总结

第9章 桥接

- 9.1 何为桥接模式
- 9.2 何时使用桥接模式
- 9.3 创建iOS 版虚拟仿真器
- 9.4 总结

第10章 外观

- 10.1 何为外观模式
- 10.2 何时使用外观模式
- 10.3 为子系统的一组接口提供简化的接口
- 10.4 在TouchPainter 应用程序中使用外观模式
- 10.5 总结

第四部分 对象去耦

第11章 中介者

- 11.1 何为中介者模式
- 11.2 何时使用中介者模式
- 11.3 管理TouchPainter 应用程序中的视图迁移

<<Objective-C编程之道>>

11.3.1 修改迁移逻辑的困难

11.3.2 集中管理UI 交通

11.3.3 在Interface Builder 中使用CoordinatingController

11.4 总结

第12章 观察者

12.1 何为观察者模式

12.2 何时使用观察者模式

12.3 在模型?视图?控制器中使用观察者模式

12.4 在Cocoa Touch 框架中使用观察者模式

12.4.1 通知

12.4.2 键?值观察

12.5 在TouchPainter 中更新CanvasView上的线条

12.6 总结

第五部分 抽象集合

第13章 组合

13.1 何为组合模式

13.2 何时使用组合模式

13.3 理解TouchPainter 中Mark 的使用

13.4 在Cocoa Touch 框架中使用组合模式

13.5 总结

第14章 迭代器

14.1 何为迭代器模式

14.2 何时使用迭代器模式

14.3 在Cocoa Touch 框架中使用迭代器模式

14.3.1 NSEnumerator

14.3.2 基于块的枚举

14.3.3 快速枚举

14.3.4 内部枚举

14.4 遍历Scribble 的顶点

14.5 总结

第六部分 行为扩展

第15章 访问者

15.1 何为访问者模式

15.2 何时使用访问者模式

15.3 用访问者绘制TouchPainter 中的Mark

15.4 访问者的其他用途

15.5 能不能用范畴代替访问者模式

15.6 总结

第16章 装饰

16.1 何为装饰模式

16.2 何时使用装饰模式

16.3 改变对象的“外表”和“内容”

16.4 为UIImage 创建图像滤镜

16.4.1 通过真正的子类实现装饰

16.4.2 通过范畴实现装饰

16.5 总结

第17章 责任链

<<Objective-C编程之道>>

- 17.1 何为责任链模式
- 17.2 何时使用责任链模式
- 17.3 在RPG 游戏中使用责任链模式
- 17.4 总结

第七部分 算法封装

第18章 模板方法

- 18.1 何为模板方法模式
- 18.2 何时使用模板方法
- 18.3 利用模板方法制作三明治
- 18.4 保证模板方法正常工作
- 18.5 向模板方法增加额外的步骤
- 18.6 在Cocoa Touch 框架中使用模板方法
 - 18.6.1 UIView 类中的定制绘图
 - 18.6.2 Cocoa Touch 框架中的其他模板方法实现
- 18.7 总结

第19章 策略

- 19.1 何为策略模式
- 19.2 何时使用策略模式
- 19.3 在UITextField 中应用验证策略
- 19.4 总结

第20章 命令

- 20.1 何为命令模式
- 20.2 何时使用命令模式
- 20.3 在Cocoa Touch 框架中使用命令模式
 - 20.3.1 NSInvocation 对象
 - 20.3.2 NSUndoManager
- 20.4 在TouchPainter 中实现撤销与恢复
 - 20.4.1 使用NSUndoManager 实现绘图与撤销绘图
 - 20.4.2 自制绘图与撤销绘图的基础设施
 - 20.4.3 允许用户触发撤销与恢复
- 20.5 命令还能做什么
- 20.6 总结

第八部分 性能与对象访问

第21章 享元

- 21.1 何为享元模式
- 21.2 何时使用享元模式
- 21.3 创建百花池
- 21.4 总结

第22章 代理

- 22.1 何为代理模式
- 22.2 何时使用代理模式
- 22.3 用虚拟代理懒加载图像
- 22.4 在Cocoa Touch 框架中使用代理模式
- 22.5 总结

第九部分 对象状态

第23章 备忘录

- 23.1 何为备忘录模式

<<Objective-C编程之道>>

- 23.2 何时使用备忘录模式
- 23.3 在TouchPainter 中使用备忘录模式
 - 23.3.1 涂鸦图的保存
 - 23.3.2 涂鸦图的恢复
 - 23.3.3 ScribbleMemento 的设计与实现
- 23.4 Cocoa Touch 框架中的备忘录模式
- 23.5 总结

<<Objective-C编程之道>>

章节摘录

版权页：插图：生成器选择建造自己的房子的人会把工程外包给承包商。

单一承包商不能建造整个房子，他将其分解为几个部分，然后转包给几个实际的建筑商（builder），他们懂得如何将零部件组装起来。

房子由风格、颜色和尺寸各不相同的部件组成。

客户告诉承包商房子里都要有什么，然后承包商协调各房屋建筑商，决定需要做什么。

应该如何建造，建筑商就如何施工。

建房子是个复杂过程，单凭一双手就想建房子，即便可能也非常困难。

如果承包商（指导者）与懂得如何建造的建筑商相互协调，这一过程将简单得多且更易管理。

有时，构建某些对象有多种不同方式。

如果这些逻辑包含在构建这些对象的类的单一方法中，构建的逻辑会非常荒唐（例如，针对各种构建需求的一大片嵌套if-else或者switch-case语句）。

如果能够把构建过程分解为客户—指导者—生成器的关系，那么过程将更容易管理与复用。

针对此类关系的设计模式称为生成器。

本章将讨论生成器模式的概念。

后面几节，也会讨论如何使用这一模式来生成RPG游戏中带有复杂特征的角色。

6.1 何为生成器模式除了客户与其所要的产品，生成器模式还包含两个重要角色：Director（指导者）和Builder（生成器）。

Builder知道究竟如何在缺少某些特定信息的情况下建造产品（什么）。

Director知道Builder应该建造什么，以参数向其提供缺少的信息来建造特定产品。

什么与如何有点儿难懂。

尽管Director知道Builder应该建造什么，这并不意味着Director知道具体Builder究竟是什么。

它们的静态关系如图6-1中的类图所示。

Builder是一个抽象接口，声明了一个buildpart方法，该builder方法由ConcreteBuilder实现，以构造实际产品（Product）。

<<Objective-C编程之道>>

媒体关注与评论

“ 每学习一门新的编程语言。
我都会去买一本介绍其设计模式的书。
这些书从来没有让我失望过。
从长远来看，我获得的回报十倍于我的付出，本书也不例外。
作者首先介绍每一种设计模式的理论和方法。
然后给出代码示例。
简单地讲，本书结构清晰，易于理解。
物超所值。

” ——亚马逊读者评论 “ 这是一本启人深思的书。
在学习如何将设计模式应用于复杂的IOS应用的同时，我开始静下来思考怎样优化既有代码。
向每一位中高级iOS开发人员推荐本书。

” ——亚马逊读者评论

<<Objective-C编程之道>>

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>