

<<设计模式>>

图书基本信息

书名：<<设计模式>>

13位ISBN编号：9787121075070

10位ISBN编号：7121075075

出版时间：2009-1

出版时间：电子工业出版社

作者：王翔

页数：652

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>

<<设计模式>>

前言

如果你要开发一个小型的系统，整个系统只有两三人，系统活不过五年，商业逻辑单纯，程序代码不超过万行，那么你随便做，影响不大。

反之，若要架构大型系统，你需要慎思，套用模型与架构，将前人的经验当作基石，这样系统设计才不至于陷入发散。

然而，你不会一开始就做大系统，那样风险太高。

因此，要练习，最好从小系统开始使用模型与架构，这样才能检讨与学习，日后方能在大系统中自如运用。

企业信息系统现今面临着大量的整合需求，需要提供深入的分析应用和灵活的应变流程。

但系统整合的复杂度是彼此系统复杂度的乘积，系统问的安全、弹性、效率、扩充性、可用性……彼此互相掣肘，此时，企业需要接触广、想得深、能定方向的架构师。

而熟悉设计模式是架构师的养成基础，要求对于问题的分类与解法有一定的认知。

有经验的设计者们，抽象出系统开发的原则与标准问题的设计解法，而GOF于十几年前提出的23种模式是其中的佼佼者。

但毕竟空有概念，仍难落实到你日常使用的程序语言中来。

坊间许多图书作者利用不同的程序语言，例如C++、Visual Basic、.NET、Java等，实现GOF的23种模式，配合UML的模型说明，让你可以方便地应用在自己的开发环境中。

本书的作者王翔有多年的开发经验，参与过多个千万乃至十亿行代码的大工程，他将经验融于设计模式中，以C#重新实现了GOF的模式，同时加入了新近的设计想法，如SOA与Web Services等，以及相对于其他设计模式而言较新的.NET Framework实现技术，如泛型、3.0的WCF等。

在本书中他除了正向地以C#展现多个不同用途的模式外，还提供了日后可重复验证与测试的单元测试码。

系统分析与设计是门艺术，问题的解法与何为问题是交织的，而各模式的搭配使用技巧不同，要领悟存乎一心，须要巧思与反复琢磨，方可有好的解法。

本书立意明确，除了告诉你问题的类型与解法，还提供了可以立即演绎的程序代码。

相信这本案头的工具书可以提供你一个不错的思维模式，帮你造就有弹性、能扩充、易维护的软件实体。

须要提醒你的是，抽象化的思考、封装与重用的设计精髓在心中，而不是落在纸上的程序代码，阅读此书时，不要停止在仅Copy and Paste程序代码。

<<设计模式>>

内容概要

本书基于C# 2.0的语法，试图将GOF 23中的模式以一种可工程化的公共库而非Example的方式呈现给读者。

内容包括以下7部分。

第1篇主要是概括性的介绍；第2篇创建型模式介绍通过将实例化职责委托他方对象的办法，隔离客户程序与具体类型实例化的依赖关系，保证客户程序（或者外部系统）获得期望具体类型实例的、同时不必发生直接的引用；第3篇结构型模式的重点在于如何通过灵活的体系组织不同的对象，并在此基础上完成更为复杂的类型（或者类型系统），而参与组合的各类型之间始终保持尽量松散的结构关系；第4篇行为型模式关注于应用运行过程中算法的提供和通信关系的梳理；第5篇主要介绍小颗粒度基础模式和应用案例；第6篇主要介绍应用全局的模式化的实现方法，包括现在已经被普遍应用的N层模式及某些关键性框架产品采用的“微内核”模式；第7篇主要是一些针对Web和Web Service领域的模式设计技术。

本书主要针对对C#语言和.NET Framework平台有一定了解或有一定应用经验的用户，尤其适于那些希望运用模式技术在设计和开发方面多应对些挑战的用户。

<<设计模式>>

作者简介

王翔，软件架构师，主要从事.NET、XML、公钥基础设施的开发。专注于数据（尤其是XML信息）的生产、加工、交换、提炼等过程。最近参与了一系列有关应用密码技术和PKI环境保护信息系统数据安全的项目。

最喜欢数学，平常案头总是摆一本数学练习题。

闲暇时间喜欢写作，通过发表多种技术文章与国内外同行交流各种数据应用经验。

项目间隙经常到各海滨城市徒步旅行、野外露营、出海航行、极限运动，这几年烹饪也渐渐成为个人主要爱好。

<<设计模式>>

书籍目录

导读第1篇 预备知识——发掘用C#语言进行面向对象化设计的潜力第1章 重新研读C#语言1.1 说明1.2 C#部分语法内容扩展1.2.1 命名空间 (Namespace) 1.2.2 简洁的异步通知机制——委托 (Delegate) 1.2.3 考验你的算法抽象能力——泛型 (Generics) 1.2.4 用作标签的方式扩展对象特性——属性 (Attribute) 1.2.5 用索引器简化的C#类型信息访问1.2.6 融入C#语言的迭代机制——迭代器 (Iterator) 1.3 可重载运算符 (Overloadable Operators) 与转换运算符 (Conversion Operators) 1.3.1 The Day After Someday1.3.2 用于有限的状态迭代1.3.3 操作集合1.3.4 类型的适配1.3.5 小结1.4 面向插件架构和现场部署的配置系统设计1.4.1 认识.NET Framework提供的主要配置实体类1.4.2 应用实例1.4.3 小结1.5 实现依赖注入1.5.1 背景介绍1.5.2 示例情景1.5.3 Constructor注入1.5.4 Setter注入1.5.5 接口注入1.5.6 基于Attribute实现注入——Attributer 531.5.7 小结第2章 开始每个设计模式之前2.1 new()的替代品2.2 准备一个轻量的内存Cache2.3 准备一个集中访问配置文件的Broker 642.4 Web? Not Web?

2.5 小结第2篇 创建型模式——管理并隔离对象实例的构造过程第3章 工厂&工厂方法模式3.1 简单工厂3.1.1 最简单的工厂类3.1.2 根据规格加工产品——参数化工厂3.1.3 简单工厂的局限性3.2 经典回顾3.3 解耦Concrete Factory与客户程序3.4 基于配置文件的Factory3.5 批量工厂3.5.1 开发情景3.5.2 定义产品类型容器3.5.3 定义批量工厂和产品类型容器3.5.4 增设生产指导顾问——Director 893.5.5 由Director指导的客户程序3.6 基于类型参数的Generic Factory3.7 委托工厂类型3.8 小结第4章 单件模式4.1 经典回顾4.2 线程安全的Singleton4.3 细节决定成败4.4 细颗粒度Singleton4.4.1 背景讨论4.4.2 解决桌面应用中细颗粒度Singleton问题4.4.3 解决Web应用中细颗粒度Singleton问题4.4.4 更通用的细颗粒度Singleton4.5 自动更新的Singleton4.6 参数化的Singleton4.7 跨进程的Singleton4.8 Singleton的扩展——Singleton-N.4.8.1 定义具有执行状态的抽象对象4.8.2 定义相应的Singleton-N实例集合4.8.3 在基本Singleton模式实现的框架下引入实例集合4.9 引入配置文件管理Singleton4.10 基于类型参数的Generic Singleton4.11 由工厂类型协助Singleton实例管理4.12 小结第5章 抽象工厂模式5.1 经典回顾5.2 按计划实施生产5.2.1 为抽象工厂补充类型映射器5.2.2 处理模式的硬伤5.3 定义计划与生产间的映射关系5.3.1 分析5.3.2 登记映射关系5.3.3 用TypeMapper协助工厂生产5.3.4 定义实体TypeMapper和实体工厂5.3.5 实现装配机制5.4 配置生产计划5.5 基于Delegate的生产外包5.6 小结第6章 创建者模式6.1 经典回顾6.2 异步调用的BuildUp() 1436.3 为Builder打个标签6.3.1 完成工具类6.4 具有装配/卸载能力的Builder 1506.5 看着图纸加工——登记配置文件6.5.1 把UML的对象变成XSD.6.5.2 把握梗概——删除不经常变化的内容6.5.3 映射为配置节点或配置元素6.5.4 实现实体对象6.5.5 完成流水线生产6.6 用迭代器控制流水线6.7 小结第7章 原型模式7.1 经典回顾7.2 表面模仿还是深入模仿7.2.1 概念7.2.2 制作实现克隆的工具类型7.2.3 克隆也要稍微保留点个性 7.2.4 定制并管理的克隆过程7.3 重新定义原型方法7.4 同时支持XML和二进制序列化的泛型集合类型处理7.5 小结第3篇 结构型模式——针对变化组织灵活的对象体系第8章 适配器模式8.1 说明8.2 经典回顾8.3 进一步扩展适配范围的组适配器8.4 Adapter——Adapter互联模式8.4.1 分析8.4.2 方式1：客户程序直接调度Adapter 1928.4.3 方式2：基于标准约定调度Adapter 1928.4.4 方式3：借助反射和约定完成异步调用8.5 用配置约定适配过程8.6 XML数据的专用适配机制8.7 小结第9章 桥模式9.1 说明9.2 经典回顾9.3 将复杂性进一步分解后的多级桥关系9.4 看着“图纸”造桥9.5 具有约束关系的桥9.6 小结第10章 组合模式10.1 说明10.2 经典回顾10.3 用迭代器遍历组合类型10.4 适于XML信息的组合模式10.5 小结第11章 装饰模式11.1 说明11.2 经典回顾11.3 具有自我更新特征的装饰模式11.3.1 分析11.3.2 抽象装饰接口11.3.3 抽象状态接口11.3.4 依据当前状态修改装饰11.3.5 测试验证11.4 设计Decorator与Builder协作的产物11.5 把Decorator做成标签11.5.1 更“彻底”的Attribute注入11.5.2 方式1：采用.NET平台自带的AOP机制实现11.5.3 方式2：自定义代理拦截框架方式11.5.4 进一步分析11.6 小结第12章 外观模式12.1 说明12.2 经典回顾12.3 Facade接口12.4 Remote Facade12.5 面向性能考虑的升级版Remote Facade——Data Transfer Object模式12.6 平台、开发语言无关的抽象Facade接口—

<<设计模式>>

—WSDL12.7 让使用者更加方便的Fluent Interface设计12.8 小结第13章 享元模式13.1 说明13.2 经典回顾13.3 制订共享计划13.4 综合性的共享体系——Object Pool 26413.4.1 应用背景13.4.2 总体技术结构13.4.3 抽象类型实体设计13.4.4 享元模式的典型应用——缓冲13.4.5 示例13.5 用于并行处理的线程级享元13.6 通过Delegate和队列实现异步Flyweight 27013.7 小结第14章 代理模式14.1 说明14.2 经典回顾14.3 远程访问代理14.3.1 第1步：定义远程访问的服务协议14.3.2 第2步：定义服务端的配置文件及相应的宿主程序14.3.3 第3步：生成客户端远程代理（Proxy）类型14.3.4 第4步：编写客户端程序14.4 数据访问代理14.5 对象缓存代理14.6 为代理过程增加预处理和后续处理的支持14.6.1 第1步：定义外部处理机制的抽象结构14.6.2 第2步：定义新增处理的配置信息14.6.3 第3步：定义配置解析对象14.6.4 第4步：修正服务端和代理类的处理过程14.7 小结第4篇 行为型模式——算法、控制流和通信关系的对象化处理第15章 职责链模式15.1 说明15.2 经典回顾15.3 用断点控制链式过程15.4 链式反应15.5 增加配置约束15.6 小结第16章 模板方法模式16.1 说明16.2 经典回顾16.3 满足多套模板要求16.4 方法的模板——Delegate16.5 类型的模板——Generic16.6 用配置勾勒的模板16.7 小结第17章 解释器模式17.1 说明17.2 经典回顾17.3 采用正则表达式17.4 采用字典17.5 采用XSD17.6 用XSD解释定制的业务语言17.7 小结第18章 命令模式18.1 说明18.2 经典回顾18.3 轻量级的Command——委托18.4 异步Command18.5 把Command打包18.5.1 外观模式方式18.5.2 组合模式方式18.6 把Command排队18.7 小结第19章 迭代器模式19.1 说明19.2 经典回顾19.3 C#化的Iterator 35319.4 小结第20章 中介者模式20.1 说明20.2 经典回顾20.3 基于Delegate和事件的松耦合Mediator 36220.4 根据配置动态协调通知关系20.5 SOAP Mediator 36620.6 小结第21章 备忘录模式21.1 说明21.2 经典回顾21.3 把备忘压栈21.4 Memento的序列化和持久化21.5 小结第22章 观察者模式22.1 说明22.2 经典回顾22.3 .NET内置的Observer机制——事件22.4 具有Observer的集合类型22.5 面向服务接口的Observer 39422.6 小结第23章 状态模式23.1 说明23.2 经典回顾23.3 State的序列化和持久化23.4 主动方式State23.5 触发式State23.6 涉及用户交互的状态流转23.7 用WF完成更易于编排的State23.8 小结第24章 策略模式24.1 说明24.2 经典回顾24.3 Strategy与Interpreter协作24.4 充分利用.NET Framework自带的Strategy接口24.5 动态策略24.6 小结第25章 访问者模式25.1 说明25.2 经典回顾25.3 借助反射实现Visitor 42825.4 用委托使引用关系更加松散25.5 小结第5篇 小颗粒度基础模式和应用案例——服务于细节的基础性模式第26章 成例26.1 说明26.2 Partial Class26.2.1 体现组合关系26.2.2 从多个侧面刻画类型26.2.3 与Visual Studio Add-ins的结合26.2.4 小结26.3 Gateway26.3.1 封装本地API的Gateway26.3.2 封装非C#语言访问接口26.3.3 封装通用操作方法26.3.4 小结26.4 Mapper 44326.4.1 介绍26.4.2 数据对象映射器26.4.3 小结26.5 Registry26.6 Value Object 45526.7 通用数据载体DataSet和DataTable26.8 Context 458第27章 GOF总结及应用案例27.1 GOF总结27.1.1 回顾GOF27.2 应用案例需求说明27.3 发现和分析27.4 模式应用27.4.1 已经采用的模式27.4.2 如何实现与具体数据源无关27.4.3 提供执行前后定制处理的能力27.4.4 设计一个结构更加灵活的连接串配置访问机制27.5 小结第6篇 部分架构模式——面向应用全局的模式化处理第28章 MVC模式28.1 说明28.2 模式介绍28.3 示例28.3.1 混合方式28.3.2 分解对象职责28.3.3 主动方式M/V/C28.4 小结第29章 管道—过滤器模式29.1 说明29.2 登机的管道流程29.2.1 数据源发起的推方式29.2.2 数据接收方发起的拉方式29.2.3 中介对象发起的推拉混合方式29.2.4 数据源/数据接收方分别发起的衔接方式29.3 示例29.3.1 推方式示例29.3.2 增加主动方式的Filter 50829.4 小结第30章 出版—预订模式30.1 说明30.2 生活中无处不在的“预订” 30.2.1 面向单一主题的本地观察者模式30.2.2 增加Proxy实现面向单一主题的分布式观察者模式30.2.3 用出版者集中管理预订30.2.4 面向物理环境设计更多出版预订模式30.3 示例30.3.1 数据实体模型部分30.3.2 业务实体模型部分30.3.3 具体实体对象部分30.3.4 单元测试30.4 小结第31章 Broker模式31.1 说明31.2 越来越庞杂的分布式系统交互之感31.3 示例31.4 小结第32章 消息总线模式32.1 说明32.2 用总线梳理企业系统环境32.2.1 分隔区域条件下的消息总线32.3 小结第7篇 部分Web和Web Service模式——面向服务开发中的模式化处理第33章 页面控制器模式33.1 说明33.2 用对象化思维抽象和扩展页面操作逻辑33.3 示例33.4 小结第34章 实现Web服务依赖倒置34.1 Web Service的模式化特征34.2 第一层的包装34.2.1 Aggregation34.2.2 Contain34.3 SOA环境下典型的Web Service开发方式34.4 依赖倒置原则及其在Web Service中的应用34.4.1 分

<<设计模式>>

析34.4.2 实现示例34.5 基于WCF的工程化实现34.5.1 实现示例34.5.2 借助WCF扩展服务的对象化特征34.6 小结第35章 Web服务适配器模式35.1 说明35.2 Web Service下的接口适配问题35.2.1 常规情景分析35.2.2 Web Service间连续交互适配35.2.3 实施方式35.3 采用标准Web Service类型的示例35.3.1 单纯数据XSD适配35.3.2 服务方法兼容性适配35.3.3 其他说明35.4 采用WCF的示例35.4.1 用Data Contract定义数据Schema35.4.2 定义不同的类型转换方式示例35.5 小结第36章 Web服务数据传输对象模式36.1 说明36.2 Web Service接口批量交互中的性能问题36.2.1 DTO对象36.2.2 DTO与消费者服务的对应关系36.2.3 部署设计考虑36.2.4 结构36.3 示例36.3.1 没有DTO对象的情况36.3.2 增加DTO对象的情况36.4 小结36.5 附件36.5.1 实现DTO数据装载的两种方式第37章 Web服务事件监控器模式37.1 说明37.2 如何为普通Web Service封装事件机制37.3 示例37.4 小结第38章 Web服务拦截过滤器模式38.1 说明38.2 通过拦截完成自定义特性的透明扩展38.2.1 实现方式38.2.2 .NET平台的实现技术38.3 示例138.3.1 基于ASP.NET Web Service的IHttpModule方式138.3.2 基于WCF自定义拦截方式38.4 小结附录A 面向关系数据和XML数据的领域逻辑模式A.1 说明A.2 实现业务领域逻辑的主要方法A.2.1 整体逻辑结构A.2.2 性能改进A.2.3 面向关系数据库的业务服务设计A.2.4 面向XML数据的扩展设计A.2.5 配置机制设计A.3 示例A.3.1 示例情景A.3.2 测试内容准备A.3.3 实际测试过程A.4 小结附录B 基于XML的应用建模B.1 说明B.2 世界是平的,但更是多元的B.2.1 更具扩展性的数据模型——XMLB.2.2 让非结构化数据可以被识别B.2.3 应对数据和内容的集成B.2.4 新的应用扩展B.2.5 应对语义网络的复杂性B.3 小结索引

<<设计模式>>

章节摘录

第1篇 预备知识——发掘用C#语言进行面向对象化设计的潜力 第1章 重新研读C#语言

1.1 说明 本章以工程化使用为目的，对C#和.NET Framework提供的几个平时开发时不引人注意的特征进行介绍，它们对于提高代码扩展性、灵活性很关键，务求用“很C#”的方式解决以往设计模式Example之外必须面临的一些问题。

它们主要包括：
 · Namespace（命名空间）； · Delegate（委托）； · Generics（泛型）；
 · Attribute（属性）； · Indexer（索引器）； · Iterator（迭代器）； · Overloadable Operators（可重载运算符）与Conversion Operators（转换运算符）； · Configuration（对象化配置访问）；
 由于降低类间耦合关系一直是设计上控制变化范围的常用手段之一，所以在设计模式之外补充有关用C#实现“依赖注入”（Dependency Injection）的方法。

有关设计模式在Threading（多线程）模型下实现需注意的内容，笔者将在相关章节的工程化分析部分介绍。

1.2 C#部分语法内容扩展 1.2.1 命名空间（Namespace） 尴尬的现实状况 是否有很好的命名空间规划是工程化代码与非工程化代码一个很明显的区别。

尤其对于大型的组织而言，如果涉及的产品线、项目、公共平台很多，如何通过命名空间把所有的代码资源有效地组织起来，恐怕是实施项目前要考虑的主要问题。

作为一个树形体系，最好有组织级统一的分类标准，目的很明确——用的时候能很容易找到。

做到这点并不容易，原因如下：
 · 习惯中很难改变的缩写命名：CAD很容易在声明的时候被命名为.**CAD，但事实上Design Guideline建议的是Cad，这样使用者和定义者之间就存在一些小小的错位。

在99%的情况下这个问题不会发生，因为您只要一个点，IntelliSense就帮您列出来了，另外的1%则发生在后绑定调用的情况下。

· 不统一的命名：涉及加密的库，可能A被定义为Encrypt，B被定义为Crypto或Cryptography。来自匈牙利命名法的“遗毒”也常常成为新旧开发人员统一命名的障碍，而且会直接影响到命名空间的定义。

· 组织或您上司的认同：技术总监不认为命名规范是他需要关心的事情，下面的架构师更关心的是结构，再下面的项目经理关注的是资源调度和进度，具体的实施人员恐怕没有多少机会规定其他同事该怎么命名，那么谁来关心命名空间呢？

企业.NET类型系统的命名空间规划示例 无论如何，即便没有办法在组织级统一命名空间，为了您所带领的团队现在做的工作在以后能更容易地被应用，或者仅仅为您自己的职业生涯好好“储蓄”，在动笔编写第一行程序之前，先规划好命名空间吧。

<<设计模式>>

媒体关注与评论

本书立意明确，除了告诉你问题的类型与解法，还提供了可以立即演绎的程序代码，相信这本案头的工具书可以提供你一个不错的思维模式，帮你造就有弹性、能扩充、易维护的软件实体。

胡百敬 微软MVP，台湾恒逸资讯资深讲师，“数据库铁人” 作者从GOF23种经典设计模式开始，带你走进模式的失门，小到细粒度的基础模式，大到粗粒度的架构模式，本书都做了详尽的讲解。

如果您还在为了软件需求的无尽变化而烦恼不断，为了在软件设计领域更上一层楼而苦苦思索，希望本书能够带给您一些启发。

李会军 微软MVP，博客园专家，IT168专栏作者 本书很有特色的地方，就是以工程角度来阐释模式，相较纯粹的模式之说，则更具普遍的下手角度，C 语言的高级特性结合设计模式的经典思想，两者相得益彰。

王涛 微软MVP，博客园专家，《你必须知道的.NET》作者

<<设计模式>>

编辑推荐

专家推荐 本书立意明确，除了告诉你问题的类型与解法，还提供了可以立即演绎的程序代码，相信这本案头的工具书可以提供你一个不错的思维模式，帮你造就有弹性、能扩充、易维护的软件实体。

胡百敬 微软MVP，台湾恒逸资讯资深讲师，“数据库铁人” 作者从GOF23种经典设计模式开始，带你走进模式的失门，小到细粒度的基础模式，大到粗粒度的架构模式，本书都做了详尽的讲解。

如果您还在为了软件需求的无尽变化而烦恼不断，为了在软件设计领域更上一层楼而苦苦思索，希望本书能够带给您一些启发。

李会军 微软MVP，博客园专家，IT168专栏作者 本书很有特色的地方，就是以工程角度来阐释模式，相较纯粹的模式之说，则更具普遍的下手角度，C 语言的高级特性结合设计模式的经典思想，两者相得益彰。

王涛 微软MVP，博客园专家，《你必须知道的.NET》作者

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>