Broadview®
www.broadview.com.cn 博文视点

设计模式

基于C#的工程化实现及扩展

王 翔 著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

设计模式

基于C#的工程化实现及扩展

王 翔 著

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内 容 简 介

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。

图书在版编目（CIP）数据

设计模式：基于C#的工程化实现及扩展 / 王翔著.—北京：电子工业出版社，2009.1

ISBN 978-7-121-07507-0

I. 设… II. 王… III. C语言—程序设计 IV. TP312

中国版本图书馆CIP数据核字（2008）第154114号

责任编辑：陈元玉
印　　刷：北京智力达印刷有限公司
装　　订：三河市皇庄路通装订厂
出版发行：电子工业出版社
　　北京市海淀区万寿路173信箱　邮编　100036
开　　本：787×1092　1/16　印张：43.5　字数：850千字
印　　次：2009年1月第1次印刷
印　　数：4 000册　　定　　价：98.00元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。

服务热线：（010）88258888。

谨以此书献给我最敬重的爷爷、献给养育我成长的父母，以及各位家人和一直以来关心、帮助和支持我的各位朋友。

联系博文视点

您可以通过如下方式与本书的出版方取得联系。

读者信箱：*reader@broadview.com.cn*

投稿信箱：*bvtougao@gmail.com*

北京博文视点资讯有限公司（武汉分部）

湖北省 武汉市 洪山区 吴家湾 邮科院路特 1 号 湖北信息产业科技大厦 1402 室

邮政编码：430074

电　　话：027-87690813

传　　真：027-87690595

若您希望参加博文视点的有奖读者调查，或对写作和翻译感兴趣，欢迎您访问：*http://bv.csdn.net*

关于本书的勘误、资源下载及博文视点的最新书讯，欢迎您访问博文视点官方博客：*http://blog.csdn.net/bvbook*

作者访谈录

博文小林　　王　翔

针对王翔老师的新书《设计模式——基于C#的工程化实现及扩展》的出版，博文视点编辑小林对他进行了邮件专访，现将博文的编辑与王老师的对话整理成文，以飨读者。

博文小林

王翔老师，您好！您即将出版的《设计模式——基于 C#的工程化实现及扩展》这本新书，是您结合项目经验撰写而成的。市面上已经有一些关于 23 种设计模式的书，有的已经获得了市场和读者的认可，您在文中重新介绍这 23 种模式时融入了哪些新元素，这些元素对读者会有哪些帮助？

王　翔

设计模式是套思想，我是一名从事开发的人员，有时候会觉得如何更巧妙地结合应用功能完成实现须要多多斟酌。

在我的师傅、同事和领导的帮助下，这几年做了一些.NET 的项目，我酷爱 C#语言，在使用中发现 C#实现一些设计模式的时候有很多特色。

虽然本书很多模式的扩展都很有限，但我希望将其作为一个引子，能够与喜欢模式 & C#语言的朋友有个讨论的基础，不断完善这部分内容。

从以上写作目的出发，我主要希望该书能够对实际工作有以下帮助：

1. 打破一些固有的套路。

2. 用 C#以简洁、直接的手段解决易于变化的问题。
3. 不要仅仅将依赖关系定格在对象体系上，应更多考虑到应用开发、运维不同生命周期中参与者的工作特点，将依赖拓宽到对象、配置体系、数据存储和服务体系。
4. 面向 Web、面向混合信息体系、面向服务。

博文小林

每个程序员都是有独立性格的个体，这些性格会给学习带来方法和思路上的影响，您认为程序员学习和使用设计模式来进行开发的时候最须关注的要点是什么？

王　翔

同为程序开发人员，我们在对待自己工作的时候总或多或少有些“止于至善”的心结。

代码、类库、应用框架不仅仅是老板和项目经理眼中的产品，更是我们敝帚自珍的工作成果，虽然我们可以接受“唯一不变的就是变化本身”，但修改自己的代码，尤其是因为上游需求不确定带来这种压力的时候，总不是愉快的。

我们要借鉴并应用那些成熟的套路，将变化抽象并集中在几个点，然后把它们交给运维人员来处理，而我们把时间更多地放在创造性的工作上。

所以，模式是现成的，但实现套路要靠自己。

如果说开发中我们用什么态度使用模式最重要，我自己的体会有两点：

1. 敏思（不唯套路，同时不囿于局部）。
2. 厉行（如果在适应变化、根据上下文需要应用某些或某个模式的时候，能够用语法支持的特性、FCL 完成，则绝不随便构造一整套类型系统，而是直达目的）。

博文小林

《设计模式——基于C#的工程化实现及扩展》一书中，我们看到了您多年经验的积累和总结。那么对于一个设计模式的初学者，您对他（她）们在学习设计模式和本书时有什么好的建议？在遇到难以捉摸的问题时有什么比较通用且有效的解决办法？

王　翔

我是个比较循规蹈矩的人，所以我的建议可能收效会慢点。我认为设计模式虽然是

思想，但需要对一个面向对象语言有比较扎实的基础。

以我自己为例。

接触.NET 是因为有个项目我师傅觉得用.NET 来实现不错，然后我去师傅那里抱佛脚，师傅打开.NET Framework SDK Documentation，指着语言参考的 C#部分和 Programming with.NET Framework 说：“这是你要的”。

不知道是不是有意指点我，记得那段时间他的 MSN Personal Message 是“聪明人懂得下笨功夫”。

既然师傅这么安排，那我就一头扎进.NET SDK，7 个月后.NET 项目延期了，不过我把那些文档全部看完了，C#语言参考还看了两遍，所有的示例代码也都调试过了。

然后看到《设计模式：可复用面向对象软件的基础》，第一遍看的时候，经常觉得“哦，这个太 Cool，有收获”，但第二遍看的时候，有很多地方会觉得“这么麻烦，用 C#，一下子就 OK 了”。

所以，设计模式是思想性的东西，但需要在语言上多做些准备，学习的时候要敢于否定自己以前已经很熟悉的套路，甚至是经典中提出的套路，自己给自己一个不断突破的目标，经过批评和自我批评之后再看《设计模式：可复用面向对象软件的基础》，应该就有更多心得和收获了。

至于难点，我师傅也有句话——“重复的力量”，看不清楚就敲击键盘调试它，几遍之后会突然有一个顿悟的时刻。本书概念上最难的是 Visitor 模式，前面在 Observer 和 Proxy 也有个跳跃式的过程，您也可以用“重复的力量”击溃它。

博文小林

您在全国海关信息中心从事过很多大型的开发项目，您日常负责的工作主要是哪些？本书重视设计模式的工程化实现和扩展，这些工程化的内容与您的工作有哪些联系？在书中您是如何体现工程化思想的？

王　翔

我的工作职责主要有三个部分：

1. 作为开发部门的高级技术架构师，主要负责行业业务系统的开发。
2. 作为信息安全工作组的成员，参与公钥项目的实施和扩展。
3. 作为优化小组的成员，参与行业主要业务系统的优化，务求 Do more with less。

本书涉及的内容基本上都是从这几年工作经历中抽象、简化出来的，虽然书名有

“工程化”，但其实更偏重“工程化实施的功能特性”，因为容错、保密、异常处理、高可用、锁和并发管理基本上没有体现在示例中。

不过本书中也提到了如何透明地加入这些机制，并且还介绍了可以即插即用的模式，希望您不要错过。

我工作的行业是一个快速变化的行业，最近几年每年的增长率都在 20%以上，同时又随着全球化、区域化而快速变化。因为业务上的变化要求，技术上必须做出相应的处理，因此发现《设计模式：可复用面向对象软件的基础》后，我就再也没有放弃过它。

这几年行业的标准化程度有了质的跨越，外部环境中 Web 2.0、SOA、云计算和纳米计算的概念也在兴起之中，所以书中很多地方提到的 XML 方式，既是顺应需要，又是不得已为之。希望其中一些粗浅的方法也能用在别的行业中。

博文小林

在经典的 23 种设计模式之外，您在书中还扩展了一些架构模式如 Web Service 模式，介绍这些并不属于经典模式的模式，您是出于哪方面的考虑？在深入这些模式之前需要有哪些必要的知识准备？

王 翔

项目需要，估计别人的项目中也已经越来越多地涉及了这些内容。

完成一个项目，不同类型的模式可能要兼容并包，“尺有所短，寸有所长”。

我有一点体会，不管什么类型的模式，哪怕它叫“架构”，都应该有个“作业面”（借鉴地质、能源行业常用的词），每次经过抽象和简化，不管什么模式其实面对的都是一个相对有限的小场景，头脑中保存的不是按层次分类的模式，而是一个长长的列表，然后根据上下文选择合适的，不要受架构层次一定要用所谓的架构模式、类层次一定要用《设计模式：可复用面向对象软件的基础》中那些对象关系的羁绊，基于组件化、服务化，我们已经可以较容易地拿捏出这个“作业面”。

至于这些扩展部分，准备知识恐怕还是经典的 GOF23 和相关领域的开发体会。

另外，比较遗憾，没能在本书中把数据访问模式、集成模式、公钥体系中的信息安全模式、XML 设计模式和数据库设计模式涵盖进来，如果有幸再版，我希望可以把它们补充进来。

推荐序 1

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，系统间的安全、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、封装与重用的设计神髓在心中，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。

微软 MVP，台湾恒逸资讯资深讲师，数据库铁人　胡百敬

推荐序 2

且看《笑傲江湖》中风清扬的独孤九剑：有进无退，招招都是进攻，使攻敌不得不守。虽只一剑九式，却是变化无穷，学到后来，前后式融会贯通，更是威力大增。能料到他要出甚么招，反招却抢在他头里。敌人手还没提起，你长剑已指向他的要害，他再快也没你快。“料敌机先”和“活学活用”这八个字，正是这套剑法的精要所在。

设计模式可以当做软件开发中的独孤九剑。在软件设计中最大的敌人就是需求不断在变化，需求变化无休无止，软件交付日期也就无限期地延迟。我们无法做到以不变应万变，但如果能提前预见到一些变化，就能用很小的代价来应对剧烈的变化。GOF 总结的经典设计模式虽只有 23 种，但不管是创建型模式、结构型模式还是行为型模式，归根结底都是在寻找软件中可能的变化，并封装这些变化。“料敌机先”为设计模式精髓之一，只不过这里的敌是需求的变化而已。

预测到了变化，我们需要运用抽象的手段对其进行封装，如何抽象、如何封装需要具体问题具体分析，不能一概而论，从重构到模式是目前使用设计模式最好的方式。对于设计模式如果不能够做到灵活自如地运用，不仅威力大减，甚至于弄巧成拙，“活学活用”为设计模式精髓之二。

然而，仅仅学会了剑法永远也无法达到武功的最高境界，正如学会了设计模式也无法登上软件设计领域之巅，要经过大量的实战才行，在实战中提高剑法，在实战中体会如何“料敌机先”，如何“活学活用”。《设计模式——基于 C#的工程化实现及扩展》正是这样一本教你进行设计模式实战的好书，作者从 GOF 23 种经典设计模式开始，带你走进模式的大门，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，希望本书能带给您一些启发。

最后，特别感谢王翔为大家带来了这样一本设计模式的经典之作。

微软 MVP，博客园专家，IT168 专栏作者　李会军

序　言

我有幸 5 岁开始学习编程。成为一名软件开发人员是从初中开始确立的一个目标。大学毕业后因为工作的关系，开始使用 Visual C++、Visual Basic 进行开发，并在师傅的教导下学习 C#。初识 C#的时候我总会将它和之前接触的语言进行类比，而且一直用 C#以很生硬的方式完成工作任务。不过随着行业的快速发展，来自各方面的变化往往在项目中期就不期而至，在很被动地完成几个项目后，我开始寻找尽可能灵活应付这些问题的方法，Enterprise Library、《Design Patterns: Elements of Reusable Object-Oriented Software》和《Patterns of Enterprise Application Architecture》是对我帮助最大的三项资料。通过对它们的了解，我发现 C#中充满灵性的内容——托管环境下的自由、灵活，我变得酷爱 C#。

模式是一个非常有趣的话题，它是对特定前提下重复出现问题的一个普遍解答，它是一种思想，使用得当也会对设计、实施提供帮助，从这个角度看它又是实实在在的生产率。最近几年单位用.NET 开发的项目越来越多，规模也越来越大，自己经常感觉到需要把一些内容记录下来，并在与同事、同行分享的过程中修正、提炼它们，这也是写本书的主要动力。区别于其他类似的图书，本书强调面向工程化处理，偏重具体实现，同时结合越来越普遍使用的 XML 技术及.NET 3.0+的技术进行了扩展和完善。在完成 GOF4 的 23 种模式后，我感觉到仅仅用这些“搭”项目是不够的，因此继续把一些架构模式、Web 服务模式、成例补充进去。不过比较遗憾的是因为时间的关系，忽略了数据库模式、数据访问模式、集成模式。

您可以直接用示例代码做练习，关于本书的示例代码您可通过以下链接免费下载：*http://bv.csdn.net/resource/sjms.rar*。为了便于了解每个知识点，建议您使用 TestDriven.Net 逐个运行相关章节示例代码的单元测试，涉及数据库访问的时候，您还需要用到微软的 Northwind 官方示例数据库。

希望本书能对您的开发有所帮助，当面对各种“不可抗拒”的变化时，您可以从中获得一些启发，能够简洁并直接地应对它们。另外，希望有机会与您就模式和 C#语言进行沟通和讨论，书中存在的问题和错误也请您不吝指正。

高级架构师　王　翔

导　读

本书的应用背景

面向对象设计模式，也就是本书简称的“设计模式”，是软件设计过程中，通过面向对象的方法对相近似的问题，在指定上下文范围内给予的指导性解决方案。模式的主要价值在于它们是以往经验的浓缩，尤其在我们建立复杂系统的时候，借鉴和采用模式可以让我们少走弯路，其设计比较灵活并具有不错的扩展性。

和 15 年前相比，现在的开发工作更强调对于业务变化的适应性。虽然有各种方法学帮助我们以尽量小的代价适应这些变化，但相信没有多少人愿意对一个已经基本完成的系统进行结构性设计修改，即便修改也希望尽可能地集中在一个点上。但现实的情况总和我们作对：业务实体这个本应该相对稳定的对象，在信息化快速建设过程中被赋予了多变的特质；业务流程和操作功能总是进行着“家常”式的变化；为了适应更广泛的服务对象，我们开发的产品需要不断集成更多的第三方产品、需要支持更多的 IT 产品；最后，还有会更加多变的未来。技术上虽然各种开发方法、架构技术都试图让 80%的开发人员仅仅关注业务逻辑的实现，但很多情况下这都是美好的愿景。简言之，开发人员处在一个夹缝之中，如果不尽量让自己的设计更具弹性，则很容易让本已经满负荷的工作不断加码。

对于准一次性代码而言，应用设计模式常常会成为负担；但对于公共库、公共平台、领域通用软件而言，合理使用设计模式则是避免“坏”设计的一个有效途径：

- 避免僵化。
- 增加重用性。
- 以适度的复杂性应对可预见的变化。

因此，如果您要“坚守”某一块代码，而且该代码总是受到打压不得不适应各种变化的话，您可以在抽象的基础上发掘变化的诱因，如果需要，参照贴近的模式设计并实现。那么为什么基于.NET Framework 呢？因为.NET Framework 已经被 MySpace 等众多成功的电子商务站点所验证，它完全可以支撑大型应用运行维护；至于选择 C#而不是 Java 和 C++，也许是因为 C#语言更优美吧。

谁应该读本书

本书的目标读者是对 C#语言和.NET Framework 平台有一定了解或应用经验的用户，尤其适于那些希望能基于模式技术在设计和开发方面应对更多挑战的用户。对于具有多年项目经验的架构师而言，本书的内容可能有些肤浅，不过您可以把本书当成一个小备忘录。

本书具体读者对象如下：

- 初学设计模式的读者

本书对于您可能有一个小的跨度，不过这并不影响您使用本书。您可以在对 C#语言语法了解的基础上，在第 1 章中补充一些 C#面向对象开发的高级知识，第 1 章中的依赖注入部分则可以先跳过。第 2 章还会为您补充一些具体的开发专题，您可以结合自己项目的需要做一套类似的小工具类型。

接着，创建型模式、结构型模式可以作为您树立并强化设计模式思想的途径。

- 中级.NET 开发工程师

您已经具有了 C#开发的经验，这时候不妨浏览一下第 1 章的内容，因为它除了进一步强化 C#语言对象化特性外，还有一些规范性、扩展性的内容。然后您可以继续学习 GOF23。建议您在学习 GOF23 后，不妨在第 26、27 章稍作停顿，回顾并对之前的了解做汇总和检查，最好结合项目中一个比较核心的类库（例如：数据访问、报文交接、通知机制……）设计一个自己的 Show case，环境可以考虑得复杂些，通过三四个迭代的开发，从中体会出 GOF23 种主要模式的适用环境。

接着，您可以根据工作的需要，继续学习后面的架构、Web Service 模式。

- 高级.NET 开发工程师

您已经在.NET 平台完成了一批规模化的项目，这时候您可以浏览本书前面的部分，不过在学习后面的架构模式和 Web Service 模式前，您不妨在第 1 章（依赖注入部分）、第 4 章、第 11 章、第 12 章、第 14 章、第 17 章、第 22 章、第 24 章和第 25 章的扩展部分稍微多花些时间，因为后续的内容很大程度上与这些章节有关系，而且它们通常在开发技巧上进行了提炼和延伸。

- .NET 平台架构师

也许您可以从附录的两章入手，其中一章介绍了领域逻辑的实现技巧（RDBMS / XML DB），另一章介绍了 XML 应用建模。推荐这两章的目的主要是技术平台，尤其是企业信息本身的变化可能需要我们思路上有些变化。至于前面 GOF23 的内容，您可能已经烂熟于心，可以先跳过。不过如果有时间，您不妨对第 11 章、第 14 章、第 17 章和第 22 章中的扩展部分稍微留意一下。接下来，您就可以直接学习后面的架构模式、Web Service 模式了。

阅读本书需要的基础知识

Example 就够了么

设计模式是一种设计思想，表达这种思想最简洁的方式就是类图，至于说明其实现上的技巧，Example 就够了。但 Demo 和实际工程应用还是有一段差距的，原因不多，但每一个都需要在 Example 之余好好考虑：

- 数据类型是 int、string，还是 DateTime？或者是<T>？
- 能支撑负载要求的并发么？
- 是不是应该用 Delegate 解决常规的异步调用？
- 如何进行运行维护？哪些允许被配置？
- 静态类、匿名方法，还有基于 Attribute 的开发是不是也可以用于实现设计模式呢？

此外，设计模式的一个亮点就是提高代码的可重用性，如果设计一套比较适合实际工程使用的设计模式库，可以重复八股式反复 Example 的工作。

设计原则

作为面向对象基本设计原则的忠实体现，设计模式帮助我们在学习过程中不断强化以下五项原则性设计要求。

- 单一职责原则（SRP）：一个类应该有且仅有一个引起变化的因素。
- 开放封闭原则（OCP）：对扩展开放，对修改封闭。
- Liskov 替换原则（LSP）：子类可以替换为它的基类。
- 依赖倒置原则（DIP）：高层模块不应该依赖于低层模块，二者都应该依赖于抽象。抽象不应该依赖于细节。细节应该依赖于抽象。
- 接口隔离原则（ISP）：一个类对另外一个类的依赖建立在最小的接口上。

而在工程上，SRP 和 ISP 常常被不经意破坏。因为设计时变化因素还没有被真正识别，因此最初设计的接口从最终的实现看，本身是可以分解的；另一个考虑就是为了开发“省事”，参数固定为某个接口，即使以后该接口被丰富了，也不需要修改下游代码。本书由于是专门描述每个设计模式工程化实现的，不涉及上层具体业务处理，因此设计上严格贯彻这五项原则，Test Project 设计也依据这五项原则展开。

约定

本书示例全部采用 VSTS 2005 编写，运行平台为.NET Framework 3.0，对于模式的每个实现的测试均采用 VSTS Test Project 的 Unit Test 方式编写。由于 VSTS 2005 自带的 Class

Designer 对 UML 的展示相对很有限，因此笔者选择用 IBM Rational Rose 或 Sparx 的 Enterprise Architect 绘制 UML。本书假设了一个叫“MarvellousWorks”的公司，并按照层次关系设定所有实例的根命名空间为 MarvellousWorks.PracticalPattern。此外，示例编码上区别于其他设计模式图书有如下不同。

- 命名规范上依照的是《Design Guidelines for Developing Class Libraries》(.NET Framework 2.0 & 3.0，本书简称为 Design Guideline)，而不是 Java 命名法、C++匈牙利命名法。
- 笔者更倾向于用 Property，而不是一对 set / get 方法。
- 对于集合类型基于 key 的检索，笔者喜欢用 Indexer 而不是类似 GetValue(TKey)的方法。
- 为了减少使用者的编码量，笔者喜欢将修饰性的类设计为 Attribute，而不是单纯的 Interface。模式实现上一般也采用 Interface<T>到 Abstract Class<T>再到 Concrete Class 的方式，务求在保持抽象性的基础上尽量减少子类实现的编码量。
- 如果某些机制的定义已经在.NET Framework 中提供了，那么尽量用平台自己的。
- 与扩展无关的、与 Assembly 内部其他类调用无关的属性、方法等一律声明为 private。
- 此外，由于本书很多介绍都是基于代码的说明，为了尽量减少行文中代码的行数，许多接口声明、方法、属性的写法都采用了很不规范的单行书写方式，例如：

C#

```
using System;
namespace MarvellousWorks.PracticalPattern.Concept.Generics
{
    public interface ITarget { void Request();}
    public interface IAdaptee { void SpecifiedRequest();}

    public abstract class GenericAdapterBase<T> : ITarget
        where T : IAdaptee, new()
    {
        public virtual void Request()
        {
            new T().SpecifiedRequest();
        }
    }
}
```

如果您经常负责代码复查，并且对代码书写方式很敏感，笔者在此对您可能会感到的不畅致歉。

相对于其他介绍设计模式思想的图书，本书重点在于如何借助.NET Framework 2.0 & 3.0 平台和 C# 2.0 实现这些模式，并且根据笔者在工程中遇到的一些情况，介绍如何扩展实现体系。

本书如何组织

本书主要基于 C# 2.0 的语法（但在部分内容上采用.NET 3.0 扩展的部分新增语法），试

图将 GOF 23 种模式以一种可工程化的公共库而非 Example 的方式呈现给读者。内容划分为 8 个部分展开。

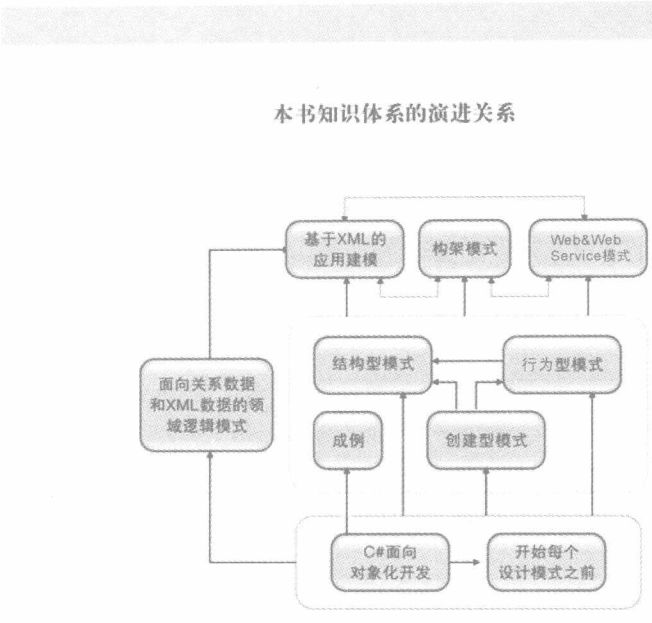

图 1

“C#面向对象化开发和扩展”是本书讨论的内容基础。图 1 中不同知识领域的技术层次以其在图中位置的高低表示，知识领域间的学习曲线关系以箭头方向而定（但部分章节内的细节内容可能有交叉）。

另外，XML 技术作为 SOA、Web 2.0 时代的主要数据形式、实体结构、处理逻辑、语义信息、领域语言载体，一直是本书所倡导和推荐的，因此“面向关系数据和 XML 数据的领域逻辑模式”及最终的“基于 XML 的应用建模”在“C#面向对象化开发和扩展”之上，贯穿全书模式介绍的始终。

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 篇主要对本书进行一些概括性介绍。

第 1 章　重新研读 C#语言

虽然本书不是一本介绍 C#语言的书，但是由于工程中，特别是规模相对较大的工程中常常需要使用一些 C#语言特有的高级特性，这些特性的介绍又要结合面向对象开发基础之