HZ BOOKS

Thinking in Patterns with Delphi

Borland
核心技术丛书

Delphi

Delphi 模式编程

刘艺 著

用最通俗易懂的语言和最明白实用的范例
解说最深奥睿智的设计模式

机械工业出版社
China Machine Press

Borland 核心技术丛书

Delphi 模式编程

刘 艺 著

机械工业出版社
China Machine Press

本书是一本供 Delphi 程序员使用的模式入门和实践读物。本书用最通俗易懂的语言和最明白实用的范例解说深奥睿智的经典著作《设计模式：可复用面向对象软件的基础》。本书不但围绕 23 个经典的设计模式进行解说，介绍其结构和用法，并给出了模式编程的实践范例；还进一步阐述了模式的概念，深入讨论了 Delphi 的模式编程机制和模式编程法则。

本书适合有编程经验的 Delphi 程序员，以及有志从事系统设计和架构、不断挑战自我发展空间的软件开发人员阅读。本书还可以作为研究生和高级开发人员的培训教材。当然也可以将此书作为一本模式编程参考手册，便于读者在项目开发中遇到实际的设计问题时直接查阅。

本书的其他相关资源和技术支持，可以在作者的个人网站 http：//www. liu-yi. net 上获得。

图书在版编目（CIP）数据

Delphi 模式编程/刘艺著．－北京：机械工业出版社，2004. 9
（Borland 核心技术丛书）
ISBN 7-111-14949-1

Ⅰ. D… Ⅱ. 刘… Ⅲ. 软件工具－程序设计 Ⅳ. TP311. 56

中国版本图书馆 CIP 数据核字（2004）第 076851 号

机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码 100037）
策划编辑：温莉芳
责任编辑：李云静
北京牛山世兴印刷厂印刷·新华书店北京发行所发行
2004 年 9 月第 1 版第 1 次印刷
787mm×1092mm 1/16·33.75 印张
印数：0001－5000 册
定价：65.00 元（附光盘）

凡购本书，如有倒页、脱页、缺页，由本社发行部调换
本社购书热线：（010）68326294

序

"Design patterns help you learn from others' successes instead of your own failures."

——Mark Johnson

《设计模式：可复用面向对象软件的基础》[1]的作者在该书的开篇感叹道："设计面向对象软件比较困难，而设计可复用的面向对象软件就更加困难"。的确，软件开发是一项极具挑战性的工作。对于编程人员而言，要做出一个良好的设计往往需要经过数次探索和反复尝试，并在大量的经验和教训之中才能找到一个较好的解决方案。尽管以编程为艺术的执着追求者们力求使面向对象设计更加灵活、优雅、健壮，但为这一目标所付出的辛苦代价同样惊人。

软件开发是一项极具挑战性的工作

好在软件业经过多年的发展已经从崇尚个人软件英雄的手工劳作时代进入了以软件工程为指导、崇尚团队合作的软件大生产的工业时代。这就是说，通过集体的贡献，一个系统可以由一些可复用的软件实体（例如组件、类等）来架构，而无需一切从头开始。

同样，在软件设计中也是如此。通过设计模式，我们可以针对一些特定的问题和场景使用一些现成的固定模式，而不必为找到一个好的方案做重新的探索。模式，实际上就是前人积累的一些宝贵经验的抽象和升华。这些宝贵的经验得以用文字等有效的形式记录下来，为我们学习和应用提供了极大的方便。

所谓模式，简单地说就是从不断重复出现的事物中发现和抽象出的规律，是解决问题的经验总结。只要是一再重复出现的事物，就可能存在某种模式。例如：小桥流水、曲径通幽的中国园林模式；柳眉杏眼、巧笑倩兮的古典美女模式；飙车枪战、美女英雄的007电影模式，等等。

只要是一再重复出现的事物，就可能存在某种模式

备受模式社区推崇的建筑学家 Christopher Alexander 说："每一个模式描述了一个在我们周围不断重复发生的问题，以及该问题的解决方案的核心。这样，你就能一次又一次地使用该方案而不必做重复劳动"。尽管他所指的是城市和建筑模式，但他的思想也同样适用于软件模式，只是在面向对象编程的解决方案里，我们用对象和接口代替了墙壁和门窗。两类模式的核心都在于提供了相关问题的解决方案。

---

[1] 《设计模式：可复用面向对象软件的基础》（《Design Patterns：Elements of Reusable Object-Oriented Software》）已由机械工业出版社翻译出版。文献引用中经常称该书的4位作者 Erich Gamma、Richard Helm、Ralph Johnson 和 John Vlissides 为 GoF，即"四人帮"。本书中也沿用 GoF 这种约定俗成的称谓。

大凡讲述软件模式的文章和书籍，总少不了把模式的源头追溯到 Christopher Alexander，广大软件开发者也因此熟悉了这位模式思想的先驱，许多人逐渐成为他的思想信徒，并把他的《建筑的永恒之道》（见图 1）和《模式语言》奉为经典。

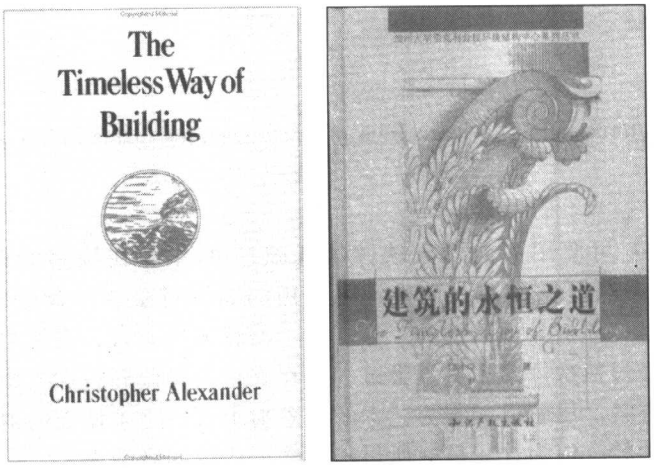


图 1　Christopher Alexander 的《建筑的永恒之道》

模式的源头在中国

其实，最早把模式的思想应用于城市和建筑中的不是 Christopher Alexander，而是中国人。模式的源头应该在中国！

早在两千多年以前，中国就形成了自己的城市建设模式，《周礼·冬官考工记第六》里记载有古代都城建设模式："匠人营国，方九里，旁三门，国中九经九纬，经涂九轨，左祖右社，面朝后市，市朝一夫。"这就是说，城市应该呈方形，每边九里，四边城墙上各设有三座城门，城内有九条直街与九条横街（或者理解为横竖各为三条街，而每条街都由三条并列的道路组成），街道之宽为车轨的九倍。城市中前面为朝廷部分（政治中心），后面为商市部分（商业中心），朝、市每边均为百步之宽。城市的左方有祖庙，右方为社稷坛。这种方整有序的城市规划一直为中国历代的封建王朝所依循并得到进一步的发展，我们从汉末三国时期的邺城、唐代的长安、宋代的汴梁一直到明清时期的北京，都可以见到这种城市模式。

北宋时期，李诫编修的我国古代建筑学经典著作《营造法式》体现了中国人在建筑模式方面的智慧。该书 1068 年开始编修，1100 年成书，1103 年刊行，历时三十多年。比 Christopher Alexander 的《建筑的永恒之道》要早 900 年。全书共 34 卷，其内容来自熟练工匠的经验，总结了官式建筑的模式和规范（见图 2 及图 3）。

十分有趣的是 GoF 在《设计模式：可复用面向对象软件的基础》一书中归纳出模式的四个基本要素为：模式名称（pattern name）、问题（problem）、解决方案（solution）、效果（consequence）；而李诫的《营造法式》则分为释名、各

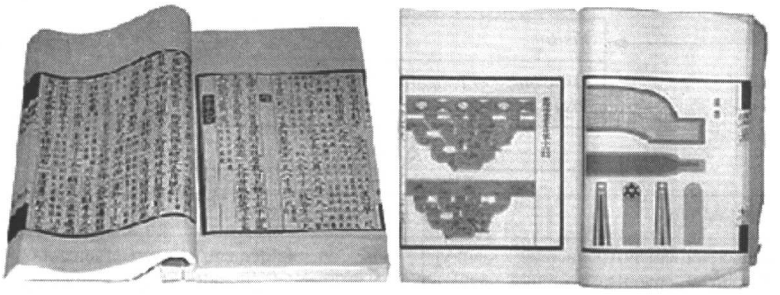


图 2　李诫编修的《营造法式》

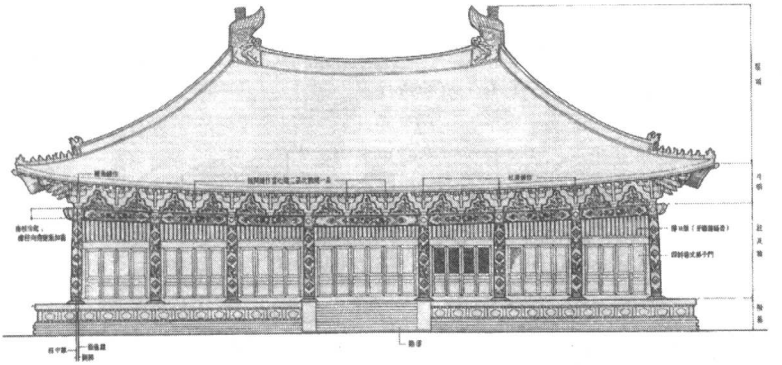


图 3　《营造法式》中的立面处理图样

作制度、功限、料例和图样五部分。通过对比，我们不难发现，其中释名相当于“模式名称”，给出模式的定义，便于交流和记忆。各作制度相当于“问题”，描述了应该在何时使用模式；它解释了设计问题和问题存在的前因后果，并可能描述了特定的设计问题。功限相当于“效果”，描述了模式应用的功效及使用模式应权衡的问题。料例和图样相当于“解决方案”，是解决所阐述问题的一个构造或配置。Christopher Alexander 强调“模式是某种场景下某个问题的解决方案”，所以这种解决方案不是抽象的，必须结合具体的“料例和图样”。料例和图样相当于《设计模式：可复用面向对象软件的基础》中的代码示例和模式的 UML 结构图（见图 4）。

模式的思维方式更符合中国式的智慧

由此可见，中国人早在古代就已经掌握了认识和应用模式的方法。模式的思维方法更符合中国式的智慧。

虽然模式的概念早就有了，但要上升到实用阶段，成为可以定义并广泛接受的模式并不简单。模式的核心就是特定的解决方案，它有效且有足够的通用性，能解决重复出现的问题。模式的另一种视角是把它看成是一组建议，而创造模式的艺术则是将很多建议分解开来，形成相互独立的组，在此基础上可以相对独立地讨论它们。

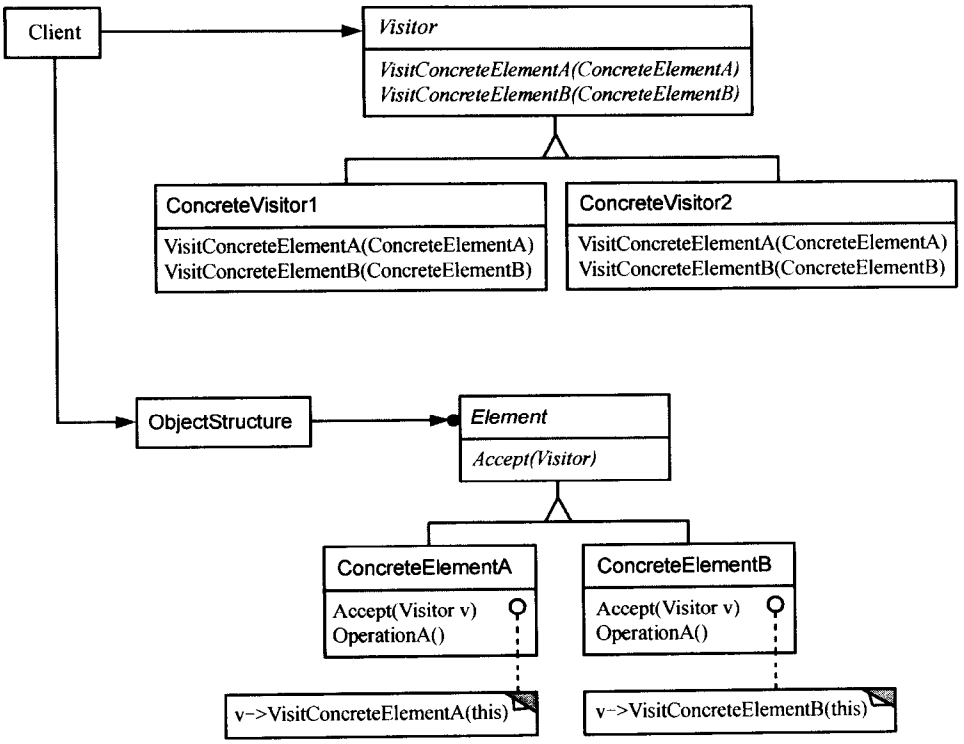


图 4 《设计模式：可复用面向对象软件的基础》中的访问者模式结构图

模式的关键在于其源于实践并指导实践

模式的关键在于其源于实践并指导实践。所以要发现模式，必须观察人们的工作过程，发现其中优秀的设计经验，并找出“这些解决方案的核心”，这并不是一个简单的过程。如果你能发现某个模式，它将是非常有价值的。你不必死背模式，甚至不必通读任何一本有关模式的书，你只需要了解到这些模式都是干什么的、它们解决什么问题、它们是如何解决问题的就足够了。这样，一旦你碰到类似问题，就可以从书中找出对应的模式来解决问题。那时，你再深入了解相应的模式也不迟。所以，很多有丰富开发经验的程序员学习模式时都会有似曾相识、相见恨晚的感觉。他们会说：“唉，原来这个问题可以这么解决！”

模式不是什么新鲜概念了。因此，模式并不是那些模式的作者人为“发明”的，而是他们从自己和别人的大量实践经验中“发现”的。他们的职责是记录通用的解决方案，找出其核心，并把最终的模式记录下来，传授给大家。对于一个高级开发人员，模式的价值并不在于它给予你一些新东西，而是在于它能帮助你更好地交流。如果你和你的同事都明白什么是门面模式，你就可以这样非常简捷地交流大量信息：“这个类是一个门面模式。”也可以对新手说：“用抽象工厂模式来解决这个问题。”模式为设计提供了一套词汇，这也是为什么模式名字如此重要的原因。

《设计模式：可复用面向对象软件的基础》作者之一的 John Vlissides 指出模式带来的四大好处是：

1）它们记录了专家的经验，并且让非专家也能理解。

2）它们的名称构成了一份词汇表，帮助开发者更好地交流。

3）它们帮助人们更快地理解一个系统——只要这个系统是用模式的方式描述的。

4）它们使系统的重组变得更容易，不管原来的系统是否以模式的方式设计。

John Vlissides 说："过去，我一直认为第一项是模式最大的好处。现在我认识到，第二条起码也同样重要。请想一想，在一个开发项目的过程中，有多少字节的信息在开发者之间流动（包括口头的和电子的）？我猜，就算没有 1GB 也有好几兆字节。（在推出了《设计模式：可复用面向对象软件的基础》之后，我已经收到了好几十兆字节给 GoF 的电子邮件。而且我们所描述的还都是小型到中型的软件开发项目。）由于有如此之大的信息交流量，所以效率上任何微小的提升都能大量节约时间。在这个意义上，模式拓宽了人们交流的带宽。我对第三、四条的评价也在逐渐提高，特别是在项目越来越大、软件生存周期越来越长的今天。至少在短期内，模式主要存在于大脑中，而不存在于工具中。如果有了方法学或自动化工具的支持，应该还有其他的收获，但是我相信这些都只是'蛋糕'上的'奶油'，而不是蛋糕本身，甚至都不能算蛋糕的一层。"

由此可见，模式并不依赖于工具或方法学的支持，自动化工具也不能使模式的应用更有效。因为模式本身就很灵活，所以你不能盲目地使用。一旦需要使用模式，你必须知道如何将它运用于当前的问题。这也是关于模式的自动化工具为什么都遭遇失败的原因。与我们经常使用的组件正好相反，实际上没有哪个模式是能让你不假思索就使用的规则或现成的程序。模式是一种"半成品"，它仅仅给出了解决方案，但并不告诉你如何去实施。为了用好它，你还必须在自己的项目中把剩下的那一半补上。我每次在使用模式时，都在不断地改动。即使同一个解决方案，也没有一次是完全相同的。虽然每个模式相对独立，但又不彼此孤立。它们有时候相互影响，有时候互相配合；在具体实践中，我们还可能将模式混合使用。例如在本书中，我在第 26 章中就给出了这样一个模式混合使用的范例。

模式并不代表模式的应用。许多自称精通模式的人，也只不过是能把 GoF 的 23 种模式背得滚瓜烂熟，并用来开开玩笑而已。我们不难在网上看到诸如：打篮球的模式、泡 mm 的模式，却很少看到模式在编程上的真正应用。

模式的困难在于模式既是一种解决问题的方法，更是解决问题的本身

应用模式要比理解模式本身更加困难。模式的困难在于模式既是一种解决问题的方法，更是解决问题的本身。因为针对同一类问题，站在不同的思考角度会有不同的模式选择。例如，解决去耦合的问题，就有用于前台对象和后台对象之间去耦合的代理（Proxy）模式；用于系统去耦合的观察者（Observer）和中介者（Mediator）模式；用于请求发送者和接收者之间去耦合的责任链（Chain of Responsibility）模式；用于抽象部分和实现部分之间去耦合的桥接（Bridge）模

式；等等。同样都是解决去耦合的问题，至于选择哪种模式，如何应用这些模式，就要看你的解决问题的出发点以及对各方面利弊的权衡了。

世界是复杂多变的，软件设计的问题在于明确目标、把握方向是件非常困难的事情。为什么说设计的本质是权衡，原因在于你很难只确定一个目标，更难确定一个固定的目标。你要考虑的往往是在多种约束下的多种可能解决方案。实际上没有哪种解决方案是唯一正确的，这就是设计策略和风格的问题。因此模式也不是万能的，模式有时仅可以看作是给你一个解决问题的建议、一个可供借鉴的经验、一个可以探索的方向。不同境界不同水平的人对模式有不同的理解，应用起来效果也不一样。

模式不是万能的，模式什么都不保证

所以那些认为模式可以保证程序更加简洁、优雅和健壮，模式可以保证软件更好的复用性的想法是幼稚可笑的。道理很简单，模式什么都不保证，模式无法取代人的位置。那些对模式“走火入魔”的人，往往对更加简单的代码编写方式视而不见，在编程中盲目使用模式。例如，本来用简单的条件表达式就足够解决的两三种不同的计算方法，却非要使用策略（Strategy）模式，反而使程序更加复杂难以理解，这无异于用大炮打蚊子。

持极端看法的人甚至认为：“水平没到，学也白学，水平到了，无师自通。所以不要学设计模式。”

这种说法虽然有一定的道理，但我并不完全同意。

模式对于初学者虽然很难，但了解和学习模式肯定比不知道模式好。尽管他们都会经过一个迷信模式、痴迷模式、滥用模式的阶段，但这不可怕，因为有招总比无招好。只要掌握正确的学习方法，活学活用，肯定要比无师自通的人少走弯路。而且随着不断的实践，在模式应用上也会更加自如。

在实践和思考中学习模式

模式正确的学习方法就是在实践和思考中学习模式，Jim Coplien 在《Software Patterns》一书中，写道：

“我更愿意用剪裁衣服的模式为这个定义打比方。我可以通过在布料上以剪裁的角度和长度来定义剪子的路径，以此告诉你如何裁一件衣服。或者，我也可以给你一个模式。你阅读剪裁说明，你可能根本就不知道在做什么，但是如果你以前做过类似的衣服，你就能完成。模式预示着产品：它既是制造产品的规则，在很多方面它又是产品本身。”

所以要学习模式，首先要有编程实践经验。对于没有编程实践经验的初学者并没有必要急于学习模式。在初学者获得一定编程实践经验之后，我认为应该让他们先掌握几种常用的模式作为解决方案，然后努力在现实中寻找使用这些方案的合适场景。不要指望能够一口气全部学完所有的 23 种设计模式。即使你博闻强记，能够学完，但如果应用得较少，过一段时间也可能会忘记。建议你通读模式书籍时先大概了解这些模式，之后根据自己的编程实践有针对性地选择那些感兴趣的模式进行实践探索。等碰到类似问题，再从书中找出对应的模式来解决问题，深入了解相应的模式。

近几年来“模式”这个词真是非常流行。就像任何流行的东西一样，GoF 的经典著作《设计模式：可复用面向对象软件的基础》一经推出就出现了洛阳纸贵、备受推崇的局面。现在模式对于软件开发的影响越来越大，国内开发人员对模式的了解虽然比国外的开发人员晚了数年的时间，但能如此引起重视的确是一件好事。不过我敢肯定，GoF 的《设计模式：可复用面向对象软件的基础》并不是一本可以轻松阅读的书。这是因为《设计模式：可复用面向对象软件的基础》整理自 Erich 的博士论文，其论文大约占了整部书的一半。对于程序员或普通读者而言，这本书过强的学术性语言和过于严谨的论述无疑增加了阅读的难度，使得这本薄薄的经典如同阳春白雪，曲高和寡。有的程序员甚至抱怨自己用一个多月的时间也啃不下这本两百多页（中译本）的薄薄小书。这也就是为什么会有那么多解释这本经典的书出现的原因。

《设计模式：可复用面向对象软件的基础》并不是一本可以轻松阅读的书

中国人向来有注疏经典的优良传统。以《周易》（易经）为例，就有朱熹的《周易本义》、王弼的《周易正义》、王夫之的《周易稗疏》、苏轼的《东坡易传》等数百种之多。同样，像《设计模式：可复用面向对象软件的基础》这样的软件业的经典著作，通过“注疏”来学习并没有什么不好。既然市面上已经有了《Java 与模式》、《Visual Basic 设计模式》这样的书籍，为什么不能有一本属于 Delphi 的模式书籍？

应该有一本能够帮助 Delphi 程序员比较全面地理解和学习模式的书籍

准确地讲，与 Delphi 有关的模式书籍我们也能找到一些。最早见到的是 ModelMaker 的《Design Patterns Reference》，里面零星介绍了一些用于 Delphi 的设计模式，这些模式仅仅当作 ModelMaker 的商业卖点介绍，而且语焉不详。其后网上广为流传的一本《设计模式：可复用面向对象软件的基础》的 Delphi 版电子书，这本书是台湾人把《设计模式：可复用面向对象软件的基础》中C ++ 的例子翻译成 Delphi 的翻版。虽然这对 Delphi 程序员学习模式有点帮助，但是有些用 Delphi 改写的程序难以运行，甚至还存有不同语言之间理解上的谬误。不久前看到李维的《Inside VCL》，但是并不像广告宣传的那样有多少与模式有关的内容，该书仅仅提到了经典模式中的 2 个，即门面模式和命令模式。而在另一本国内作者陈省的《Delphi 深度探索 第 2 版》中，虽然有一章专门介绍设计模式，但是 80 页的篇幅似乎过于简洁，而且仅限于“以 VCL 中的示例来介绍如何使用 Delphi 应用设计模式的内容”。

我觉得应该有一本能够帮助 Delphi 程序员比较全面地理解和学习模式的书籍。模式的困难在于其很强的实践性，所以写这样一本书有很大的难度。我写这本书的目的并不是为了在 GoF 的模式上做什么创新，而是为了通过通俗易懂的解说和明白实用的例子来帮助 Delphi 程序员掌握模式的使用，建立模式的思维。因此，我花费了大量的时间和精力改编和设计了实践模式的 Delphi 编程范例，其目的是为了让读者看到模式在实践应用中的价值。

可能一些程序员在阅读技术书籍时会说：“别给我们讲大道理，给我们看程序！”

我要说他们的这些要求是对的！很多关于模式的书故作高深，晦涩难懂，其示例代码却很少，让我们看不到任何可以执行的完整代码。我反对这种风格的技术写作。如果我们的目的不是传授知识、解决问题，那么何必要浪费笔墨？

用最通俗易懂的语言和最明白实用的范例解说深奥睿智的《设计模式：可复用面向对象软件的基础》

本书可以看成是供 Delphi 程序员使用的模式入门和实践读物。我的想法是力求用最通俗易懂的语言和最明白实用的范例解说深奥睿智的《设计模式：可复用面向对象软件的基础》。

为了让更多的人看懂这本书，我在每个模式的开篇都会用生活中的例子和图示进行形象的模式解说；为了便于套用模式，我为每个模式提供了 Delphi 的代码模板；为了给程序员看程序，我为每个模式精心准备了最实用的范例程序，附上完整的源代码和代码剖析，保证你能运行这些真正的模式应用程序（而不是一个空壳）。这些范例程序涉及的行业应用领域包括：酒店管理系统、银行管理系统（转账系统、信用卡管理系统）、人力资源管理系统（组织机构管理、薪酬福利管理）、地理信息系统、聊天室系统、任务调度系统、项目审批系统等。

当然，书中的范例毕竟只是经过处理的简单的例子，甚至还会有为讲解模式而应用模式之嫌，离特殊而复杂的实际应用尚有距离。就像模式什么都不保证一样，范例程序同样不能保证可以复制到你的项目中取代你自己编程，因此不要指望有太多现成的代码可以拷贝。本书只是帮你理解设计模式，告诉你这个模式是怎么一回事，大概可以处理什么样的情况，这样处理有什么好处，以及如何用 Delphi 编程实现。最终的解决方案需要你自己去权衡，模式编程需要你自己去完成。

模式只是开始，并非终点

当你使用模式时请记住，它们只是开始，并非终点。让我们这些作者去囊括项目开发中的所有变化和技术是不可能的。本书的目的也只是作为一个开始，希望它能够把我自己的和我所了解的经验与教训传递给读者，你们可以在此基础上继续努力。请读者记住，所有模式都是不完备的，如果你们决定进入模式编程的新境界，你们就有必要在自己的软件开发中实践它们、完善它们。

刘　艺

www. liu-yi. net

2004 年 7 月 2 日于南京

前　言

本书经过多年的酝酿和一年多的艰难写作终于完稿了。这本书最初仅仅是我本人学习模式的一个私人笔记，所以使用《Thinking in Patterns with Delphi》的英文名称可能更为准确。

本书第一部分“模式编程原理”，阐述了模式的概念，深入讨论了 Delphi 的模式编程机制和模式编程法则；后面各部分则分别围绕 23 个经典的设计模式进行解说，介绍其结构和用法，并给出模式编程的实践范例。

对于有编程经验的 Delphi 程序员来说，阅读本书并不困难。书中的例子他们大都很熟悉，只不过以前可能没有用模式编程的方式实现过。在比较不同的思考问题的方法和解决问题的途径后，我相信他们会有“于我心有戚戚焉”的感受。在理解模式的基础上，如果进一步深入阅读本书的第 2 章“Delphi 的模式编程机制”和第 3 章“模式编程法则”将会有更高层次的收获。

诚恳地讲，这部书并不适合初学者阅读，甚至不适合没有建立面向对象概念和不了解面向对象编程的读者阅读。不过初学者可以先积累一些编程实践经验，并通过阅读本人的《Delphi 面向对象编程思想》来建立面向对象的思维方式。然后，尝试阅读本书中一些较为简单和常用的模式，例如：工厂方法模式、策略模式等。

本书的结构是松散的，各个模式相对独立，自成一章。强烈建议读者在阅读时，先跳过那些你们认为难读的章节和暂时用不上的模式。我并不是说这些章节不重要，而是说最后再回过头来阅读这些章节效果会更好！

当然也可以将此书作为一本模式编程参考手册，便于读者在项目开发中遇到实际的设计问题时直接查阅相关章节，而不需阅读全书。

本书的光盘中包含了书中绝大多数示例程序的源代码，并在 Delphi 7 上调试通过。

本书的其他相关资源和技术支持，可以在我的个人网站和博艺论坛上获得：http://www.liu-yi.net 。另外，感兴趣的个人和单位亦可直接和我本人联系相关的培训。

由于本人水平有限，加之可能的笔误，书中难免会有疏漏之处。为此我在博艺论坛（http://www.liu-yi.net/bbs/index.asp）上开辟了《Delphi 模式编程》讨论版，欢迎大家及时把勘误意见贴在上面，以便在重印时修订。

最后要感谢邵印中为本书所做的校对工作，感谢周赛锋为本书提出了很好的建议，感谢段立、罗宾、李启元、洪蕾、吴永逸、吴英、杨德刚、王远、刘霞、尹敬湘、刘藩、刘哲雨、罗勇、杜军在本书写作中给予的支持。如果没有家人、朋友、读者的厚爱，本书可能永远无法完成。

此外，还要衷心感谢多年来不断支持我技术写作的机械工业出版社华章分社，与他们合作是令人愉快的。

刘　艺

www.liu-yi.net

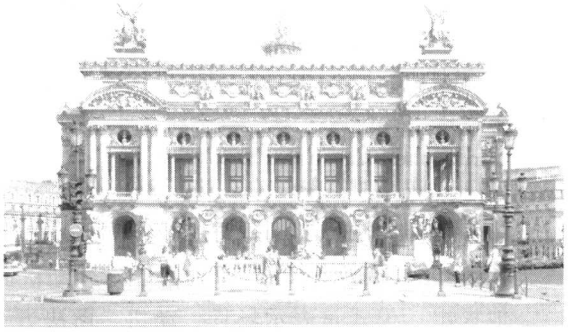
2004 年 7 月 5 日于南京

目　录

第三部分 结构型模式编程

第四部分 行为型模式编程

第一部分
模式编程原理



第1章 模式概述

1.1 模式的概念

1.1.1 什么是模式

模式(pattern)的概念最早由建筑大师 Christopher Alexander 于20世纪70年代提出，应用于建筑领域。

20世纪80年代中期由 Ward Cunningham 和 Kent Beck 将其思想引入到软件领域，1994年开始由 Hillside Group(由 Kent Beck 等发起成立)和 OOPSLA 联合发起了国际程序设计模式语言大会(Patterns Languages of Program Design，简称 PLoP 或 PLoPD)，如今模式已成为软件工程领域内的一个热门话题，其在计算机领域的影响超过了在建筑界的影响。

软件业界的模式热最早源自 Erich Gamma、Richard Helm、Ralph Johnson 和 John Vlissides[㊀]的经典著作《Design Patterns：Elements of Reusable Object-Oriented Software》[㊁]。这本书用C ++ 和 Smalltalk 语言的示例代码来阐明了23种设计模式及其实现。模式的推广和应用近年来发展十分迅速，涉及程序设计、系统架构、教学研究、组织管理等多个方面。在推动模式中起重要影响的著作还有 Frank Buschmann、Regine Meunier、Hans Rohnert、Peter Sommerlad 和 Michael Sta[㊂]合著的《Pattern-Oriented Software Architecture：A System of Patterns》[㊃]以及程序设计模式语言大会的精选论文集《Pattern Languages of Program Design》和《Pattern Languages of Program Design 2》

Christopher Alexander 说过："每一个模式描述了一个在我们周围不断重复发生的问题，以

㊀ 文献引用中经常称这4人为 GoF，即"四人帮"。本书中也采用 GoF 这种约定俗成的称谓。
㊁ 中译本《设计模式：可复用面向对象软件的基础》已由机械工业出版社翻译出版。
㊂ 文献引用中经常称这5人为 GoV，即"五人帮"。
㊃ 中译本《面向模式的软件体系结构 卷1：模式系统》已由机械工业出版社出版。