TURING 图灵程序设计丛书

Pattern-Oriented Software Architecture Volume 2

Patterns for Concurrent and Networked Objects

面向模式的软件架构

并发和联网对象模式

卷2

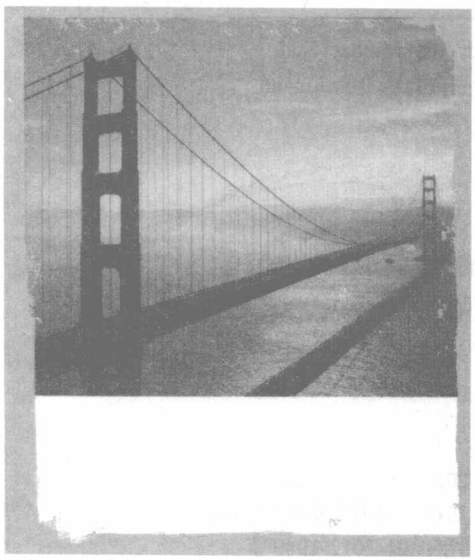
- 介绍面向对象联网与并发中间件的开山之作
- 汇聚国际公认的软件开发专家集体智慧
- 模式示例丰富，轻松解决软件架构设计问题

[美] Douglas Schmidt
[德] Michael Stal 著
[德] Hans Rohnert
[德] Frank Buschmann
朱而刚 袁国忠 译

人民邮电出版社
POSTS & TELECOM PRESS

Pattern-Oriented Software Architecture Volume 2
Patterns for Concurrent and Networked Objects

面向模式的软件架构
并发和联网对象模式



卷2

[美] Douglas Schmidt
[德] Michael Stal 著
[德] Hans Rohnert
[德] Frank Buschmann
朱而刚 袁国忠 译

人民邮电出版社
北 京

图书在版编目（CIP）数据

面向模式的软件架构. 第2卷, 并发和联网对象模式 /
(美) 施密特 (Schmidt,D.) 等著 ; 朱而刚, 袁国忠译.
-- 北京 : 人民邮电出版社, 2013.12
（图灵程序设计丛书）
书名原文: Pattern-Oriented Software
Architecture, Volume 2: Patterns for Concurrent
and Networked Objects
ISBN 978-7-115-33214-1

Ⅰ. ①面… Ⅱ. ①施… ②朱… ③袁… Ⅲ. ①软件设
计 Ⅳ. ①TP311.5

中国版本图书馆CIP数据核字(2013)第234653号

内 容 提 要

本书这一卷是介绍构建面向对象的联网与并发中间件的开山之作，揭开了构建中间件的神秘面纱。本卷以4位大师的经验为导向，讨论了16个模式和一个成例，仔细解释了常见的设计问题、驱动因素、成功的解决方案以及使用效果。

本卷面向专业的软件开发人员，尤其是开发并发与联网系统的专业软件开发人员。

◆ 著 [美] Douglas Schmidt [德] Michael Stal
[德] Hans Rohnert [德] Frank Buschmann
译 朱而刚 袁国忠
责任编辑 刘美英
执行编辑 李 静
责任印制 焦志炜

◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路11号
邮编 100164 电子邮件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京鑫正大印刷有限公司印刷

◆ 开本：800×1000 1/16
印张：27.25
字数：638千字 2013年12月第1版
印数：1-3 000册 2013年12月北京第1次印刷
著作权合同登记号 图字：01-2012-1949号

定价：89.00元

读者服务热线：(010)51095186转604 印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京崇工商广字第0021号

站在巨人的肩上

Standing on Shoulders of Giants

www.ituring.com.cn

版权声明

时遍历一个又一个模式。

作者提到，本书所探讨的一些想法与技巧是对网络编程相关书籍内容的补充，比如 W. Richard Stevens 的诸多开山之作（例如介绍网络编程的[Ste98]）。本书的主要出发点是对更高层面设计问题持续关注，例如，本书并不讨论 UNIX `select()` 调用的细节，而是阐释如何基于 `select()` 和其他操作系统调用构建一个组合式的灵活框架——Reactor。

现代平台提供了 I/O、线程、同步以及事件分离等方方面面的功能，如何应用这些功能作为构建更高层次框架和组件的基础，也是本书隐含的主题之一。本书重点关注 UNIX 与微软操作系统中的 C/C++并没有偏离上述主题。例如，Java 编程人员将会发现在某些情况下会有一些微小的不相关，这些情况包括：Java 已经直接实现了本书讨论的一些模式，比如 Scoped Locking；Java 提供了按照模式的特定实现组织的框架，比如 JavaBeans 框架对可配置组件的支持；Java 无法访问底层系统机制，比如同步事件分离。

然而，如果你非常熟悉 Java、Smalltalk 以及其他 OO 编程语言，那么仍可以从模式表达的核心思想中获益，而且可以更好地领会一些模式为什么在语言特性和库中是直接受支持的以及如何实现这些支持，还将能够基于其他模式构建出有用的组件。我们来看一个示例，直到 `java.nio` 出现，Java 才为异步 I/O 提供了访问系统组件的方法，但是在参考了本书所讲的 Proactor 模式后，我曾经完成了一个 Java 程序，通过一个简单的自循环（spin-loop）线程检测多通道上的 I/O 可用性，从而模拟了这种分离。上述实现效率虽然不高，但在它的目标使用环境中已经绰绰有余。

经过多年发展，本书所讲的一些模式，比如 Reactor，已经从对设计发明（design invention）的描述成长为设计模式。对于有构建可移植 OO 中间件经验的开发人员而言，每个人肯定都写过或使用过至少一个 Wrapper Facade。但是本书包含了其他一些早期出现的模式，并探讨了其设计方面新的贡献。模式必须是经得起时间检验、可以独立（重新）发现的解决方案，虽然起初对于这样的描述能否成为模式还有点不太确定，但是经历了时间的检验，本书所讲的模式确实抓住了关键驱动因素和设计问题的实质，作者和 OO 中间件社区对于这一点越来越有把握，他们还见证了本书介绍的解决方案在各种各样的环境中反复应用。

在这个大背景下，我邀请你来分享本书。通过阅读尤其是使用本书中的材料，读者将会发现在 OO 中间件开发人员的圈子里，Reactor 和 Proactor 等模式名称已经人尽皆知，就如同 OO GUI 开发人员圈子里的 Decorator 及 Observer 一样。

Doug Lea
于纽约州立大学奥斯威戈分校

关于本书

模式已经风靡整个软件开发社区。自从扛鼎之作《设计模式：可复用面向对象软件的基础》[GoF95]面世以来，软件开发人员对于模式满腔热忱。模式最早出现在软件成例[Cope92]、建筑结构模式[Ale79] [AIS77]以及人类学文化模式[Bat97]等相关早期作品中，接下来出版的作品，如《程序设计的模式语言》（PLoPD）系列 [PLoPD1] [PLoPD2] [PLoPD3] [PLoPD4]以及《模式系统》[POSA1][①]等，更是推动这种对模式的高涨热情更上一层楼。

本书为《面向模式的软件架构》（*Pattern-Oriented Software Architecture*，POSA）系列丛书的第2卷。同卷1《模式系统》[POSA1]类似，本书收录了诸多模式和最佳实践，由此阐释了实现高质量软件系统所需的久经考验的、具体的有用技术。书中的模式和最佳实践能够并且已经应用到各种不同的领域，包括电信和数据通信、金融服务、医疗工程、航空航天、制造工艺流程控制以及科学计算。上述这些模式和最佳实践还奠定了目前流行的分布式对象计算中间件的基础，例如CORBA [OMG98c]、COM+ [Box97]、Java RMI [WRW96]以及Jini [Sun99a]。

此外，本书所有模式同[POSA1]一样，都是建立在同样严密的概念基础之上。例如，我们使用了相同的模式分类方式和相同的模式描述格式，介绍了 C++、Java 以及 C 在内的多种编程语言方面的示例和已知应用。

因此，本书这一卷与上一卷秉承相同的思想体系和路线，具有同样的“观感”（look and feel）。

《模式系统》涵盖的通用模式范围非常广，而本书则与之相反，仅关注更为特定的领域：*并发与联网*。本书中的所有模式都以上述两个领域为中心，这样我们就能够更加深入地探讨许多并发与联网相关主题。如果本书收录很多不相关领域的模式，则无法达到这样深入的效果。在上述这些日益重要的软件开发领域中，本书中的模式对《模式系统》中的通用模式是一种补充。

但是，我们关注的是并发与网络应用程序和中间件中相关的*通用、独立于领域*的模式，目标是让本书中的模式尽可能地对你日常工作中的项目有所帮助。因此，本书并不涉及那些*局限于特定应用领域*的模式，比如[DeBr95] [Mes96] [ACGH+96]中的模式，这些模式解决网络方面的问题，但只适用于电信领域。

---

① 我们将《模式系统》简称为[POSA1]，其间并未提及作者，对本卷也这样。我们将本卷简称为[POSA2]。这样做是为了避免读者将 POSA 的某卷与某位作者尤其是第一作者联系起来。

由于只关注通用的、独立于领域的并发与联网相关模式，本书还充实了现有的并发网络编程与面向对象设计方面的相关文献。

- 并发网络编程文献通常只关注操作系统 API 的语法与语义，比如 Sockets [Ste98]、POSIX Pthreads [Lew95]或者 Win32 threads [Ric97]，这些 API 可以协调对主流操作系统上所提供的内核级通信框架的访问，比如 System V STREAMS [Ris98] [Rago93]。作为补充，本书描述了如何在高质量的并发与联网系统的设计与编程中有效使用这些 API。
- 探讨解决上层软件设计与质量要素的文献[Boo94] [Mey97] [DLF93]通常不会关注并发与联网应用程序的开发，所以填补这一空白也是本书的主题。

[POSA2]与[POSA1]的另一个不同之处在于，[POSA2]描述的模式并不只是为了组成一个模式名录或者模式系统，这些模式还互相协作，互为支撑，为并发与联网软件相关的模式语言奠定了基础。本书在介绍其他模式文献中模式的基础上，描述了这种模式语言如何应用于构建复杂的并发与联网软件系统和应用程序、Web 服务、分布式对象计算中间件以及底层操作系统的网络协议与机制。

但是，我们会分开介绍每个模式以及它们是如何组成模式语言的。首先以自成一体的结构来描述模式本身，以便使这些模式能够应用于最有用的环境中。第 1 章描述了模式之间如何交互以及如何与其他模式互为补充。

但需要注意的是，本书中的很多模式并不只应用于并发与联网环境中。为了举例说明这些模式的应用范围之广，我们介绍了其他领域中的应用，比如基于组件的软件系统或者交互式软件系统。另外，还给出了示例，说明如何在日常生活中应用这些模式。

由于某些模式的初始版本已经发表在 PLoP 系列丛书[PLoPD1] [PLoPD2] [PLoPD3] [PLoPD4]和 *C++ Report* 杂志上，所以它们可能看起来很眼熟，但是本书对这些早期版本做了大量的完善工作：

- 首次将这些模式结集出版，突出了它们所表述的模式语言；
- 基于在会议和研讨班上收到的许多改进建议（有电子邮件，也有来自内部的大量评审意见以及领头人作出的评论），我们对这些模式进行了大量的重写和修改；
- 这些模式都已转换为POSA模式格式，并且描述风格一致。

目标读者

同已出版的[POSA1]类似，本卷的目标受众是专业的软件开发人员，尤其是那些开发并发与联网系统的专业软件开发人员。本书将帮助这些专业的软件从业人员以一种崭新的方式思考软件架构，为他们在大型复杂的中间件和应用程序的设计与编程中提供支持。

如果高年级本科生或者研究生在网络和操作系统方面具有坚实的知识基础，或者想学习有效设计和实现上述系统所需的核心原理、模式及技巧，那么本书同样适用。

结构与内容

本卷可以用作教科书，从头到尾进行阅读，或者用作参考指南，从而详细地探究具体模式的细微之处。

第 1 章提纲挈领地介绍了开发人员在开发面向对象的并发与网络应用程序和中间件时所面临的挑战。我们使用了并发的 Web 服务器程序作为现实中的示例，说明了这些领域中的关键点，包括服务访问与配置、事件处理、同步以及并发。

第 2 章到第 5 章构成了本书的主体。这 4 章介绍的全部是“货真价实”的模式[U2]，汇集了开发高质量的并发与联网系统所需的行之有效的原则与技巧。当你开发自己的并发与联网应用程序或者整理发现的模式时，我们希望这些模式能够作为实用的范例。

第 6 章探讨了如何将第 2 章至第 5 章介绍的模式联系起来。我们还展示了如何将这些模式与文献中的其他模式联系在一起，为并发联网系统与中间件形成一种模式语言。前面提到，有些模式并不只能用于并发与联网系统环境。对于这样的模式，我们将概述其应用范围。

第 7 章重温了我们 1996 年关于“模式未来”的预测，该预测发表于[POSA1]。我们讨论了在过去四年里模式的实际发展方向，分析了模式以及模式社区的现状。在回顾上述内容的基础上，修正了模式以及模式语言未来研究与应用的前景。

本书最后是对所介绍模式的总结、常用术语表、符号附录、该领域相关工作的详细参考资料以及各种主题索引。

本书相关的补充材料可以在线访问 http://www.posa.uci.edu，该 URL 还提供了 ACE 与 TAO 源码的链接，包含了本书所有模式的 C++示例和一些 Java 示例的源码。

毋庸置疑，有些并发与联网对象系统的相关方面我们略过未讲，有些模式则将随时间的流逝在应用和扩展模式语言的实践过程中出现。如果有任何评论、建设性的批评或者对本书风格与内容有任何改进意见，请发送电子邮件到 patterns@mchp.siemens.de。同样欢迎公开讨论我们在模式方面所做的工作。请使用邮件列表 siemens-patterns@cs.uiuc.edu 向我们发送反馈、评论和建议。邮件列表的订阅指南可以在模式的主页上找到，其 URL 为 http://hillside.net/patterns/。该链接还提供了很多模式方面的重要信息来源，例如已经出版和即将出版的书籍、模式方面的会议信息与发表的论文等。

模式抢先看

服务访问和配置模式

Wrapper Facade 设计模式将既有的非面向对象 API 提供的函数和数据封装到面向对象的类接口中，后者更简洁、更健壮，可移植性、可维护性和内聚性更高。

Component Configurator 设计模式让应用程序能够在运行阶段加载和卸载组件实现，而无需修改、重新编译和静态地重新链接应用程序。Component Configurator 还支持将组件重新分配到其他进程中，而无需关闭并重启正在运行的进程。

Interceptor 架构模式使得可以透明地给框架添加服务，并在特定事件发生时自动触发它们。

Extension Interface 设计模式让组件能够导出多个接口，从而避免因开发人员扩展或修改既有组件的服务功能，导致接口“膨胀”及客户端代码失效。

事件处理模式

使用 Reactor 架构模式，事件驱动的应用程序可以分离并分派从一个或多个客户端发送到应用的服务请求。

Proactor 架构模式能够高效地为事件驱动应用程序分离及分派由异步操作完成触发的服务请求。该模式既提供了并发的性能优势，又不会产生不良后果。

使用 Asynchronous Completion Token 设计模式，应用程序能够高效地分离并处理异步操作的响应，这些异步操作由应用程序在服务中进行调用。

Acceptor-Connector 设计模式将网络系统中互相协作的端服务的连接与初始化，同之后的处理分离。

同步模式

无论控制代码从一个作用域的返回路径如何，Scoped Locking C++成例都能确保：当控制代码进入该作用域时，能够自动获取到锁；当控制代码离开该作用域时，能够自动释放锁。

Strategized Locking 设计模式对组件中使用的同步机制进行参数化，保护对组件临界区的并发访问。

Thread-Safe Interface 设计模式。将锁定的开销降到最低，并且确保组件内的方法调用不会导致“自我死锁”，自我死锁发生于试图重新获取组件已经持有锁的情况下。

在程序执行期间，每当代码的临界区必须以线程安全的方式，只进行一次获取锁操作时，Double-Checked Locking Optimization 设计模式可以降低竞争和同步的开销。

并发模式

Active Object 设计模式将方法执行和方法调用分离，旨在改善并发性、简化对位于独立控制线程中的对象的同步访问。

Monitor Object 设计模式同步同时调用的方法，确保每次只运行对象的一个方法。它还让对象的方法相互协调，以确定它们的执行顺序。

Half-Sync/Half-Async 架构模式将并发系统中的异步处理和同步处理分离，以简化编程工作，同时又不降低性能。它引入了两个相互通信的层，一层处理异步服务，另一层处理同步服务。

Leader/Followers 架构模式提供了一个高效的并发模型，其中多个线程轮流检测一组事件源，对其发出的服务请求进行分离、分派和处理。

Thread-Specific Storage 设计模式让多个线程能够使用相同的“逻辑全局”访问点来获取线程本地的对象，避免了每次访问对象的锁定开销。

致谢

很高兴能够向支持我们完成本书创作的一些人表示感谢，感谢你们分享知识，也感谢你们审阅本书各部分的早期手稿，并提供宝贵的反馈意见。

评审冠军的桂冠颁发给我们尊敬的同事们：Regine Meunier、Christa Schwanninger、Martin Botzler、Lutz Dominick、Prashant Jain、Michael Kircher、Karl Pröse 以及 Dietmar Schütz。我们曾经召开了无数次作者讨论会，他们在讨论会上花费了大量的宝贵时间帮助我们审阅手稿，对本书内容精雕细琢，最终使本书定稿。同样要感谢分布式对象计算（DOC）组成员 Tim Harrison、Prashant Jain、Carlos O'Ryan 以及 Irfan Pyarali，他们是本书六个模式初始版本的作者。来自慕尼黑西门子公司、圣路易斯华盛顿大学以及加州大学欧文分校的研究人员，同本书的四个主要作者一起组成了整个 POSA 写作团队。

我们还要向 Peter Sommerlad、Chris Cleeland、Kevlin Henney 以及 Paul McKenney 表示最诚挚的感谢。Peter 作为我们的管家，关注材料的正确性、完整性、一致性和质量，细致入微地审阅了所有材料。Chris、Kevlin 以及 Paul 担任我们的同行评审人，提供了更多的详细反馈。他们四人为提高本卷的质量贡献卓著。

我们还要向伊利诺伊大学香槟分校的软件架构组表示感谢，其成员包括 Federico Balaguer、

John Brant、Brian Foote、Alejandra Garrido、Peter Hatch、Ralph Johnson、Dragos Manolescu、Brian Marick、Hiroaki Nakamura、Reza Razavi、Don Roberts、Les Tyrrell、Joseph W. Yoder、Wanghong Yuan、Weerasak Witthawaskul 以及 Bosko Zivaljevic，他们就很多 POSA2 模式举办了多次作者研讨会，反馈了很多宝贵意见，帮助我们对本书内容去伪存真，使其通俗易懂。

来自世界各地的很多其他朋友反馈了对本书早期版本的宝贵意见，他们是 Giorgio Angiolini、Brad Appleton、Paul Asman、David Barkken、John Basrai、Joe Bergin、Rainer Blome、Don Box、Martina Buschmann、Tom Cargill、Chuck and Lorrie Cranor、James O. Coplien、Ward Cunningham、Gisela Ebner、Ed Fernandez、Erich Gamma、Sonja Gary、Luciano Gerber、Bob Hanmer、Neil Harrison、Michi Henning、David Holmes、Tom Jordan、Fabio Kon、Bob Laferriere、Greg Lavender、Doug Lea、John MacMillan、Mittal Monani、Duane Murphy、Jaco van der Merwe、Michael Ogg、Bill Pugh、Dirk Riehle、Linda Rising、Wolfgang Schroeder、Richard Toren、Siva Vaddepuri、John Vlissides、Roger Whitney 以及 Uwe Zdun。每章模式的“致谢”部分概述了他们为润色本书所做出的宝贵贡献。

我们万分感谢那些来自圣路易斯华盛顿大学、加州大学欧文分校、Object Computing Inc.以及 Riverace 的 DOC 组员。他们将本书所讲的所有模式具体化，完善优化应用到 ACE 及 TAO 中间件项目的组件与框架中。这些带给我们启示的组员有 Everett Anderson、Alex Arulanthu、Shawn Atkins、Darrell Brunsch、Luther Baker、Matt Braun、Chris Cleeland、Angelo Corsaro、Sergio Flores-Gaitan、Chris Gill、Pradeep Gore、Andy Gokhale、Priyanka Gontla、Myrna Harbison、Tim Harrison、Shawn Hannan、John Heitmann、Joe Hoffert、James Hu、Steve Huston、Prashant Jain、Vishal Kachroo、Ray Klefstad、Yamuna Krishnamurthy、Michael Kircher、Fred Kuhns、David Levine、Ebrahim Moshiri、Michael Moran、Sumedh Mungee、Bala Natarjan、Ossama Othman、Jeff Parsons、Kirthika Parameswaran、Krish Pathayapura、Irfan Pyarali、Carlos O'Ryan、Malcolm Spence、Marina Spivak、Naga Surendran、Selcuk Uelker、Nanbor Wang、Seth Widoff 以及 Torben Worm。我们还想向来自世界各地成千上万的 ACE 和 TAO 用户所做出的巨大贡献表示感谢。在过去的十年里，他们一直在使用并优化本书描述的模式和框架组件。没有他们的支持、持续的反馈和鼓励，也不会有本书。

特别感谢 Johannes Nierwetberg、Lothar Borrmann 以及 Monika Gonauser，感谢他们在德国慕尼黑西门子 AG 企业技术软件工程实验室所给予的管理上的帮助与支持。我们还要感谢慕尼黑西门子 AG 通信设备业务部门的 Calinel Pasteanu，感谢他对本书写作的繁杂与快速发布产品的压力的理解。

我们还要感谢同事们和赞助商们对于研究模式以及 ACE 与 TAO 中间件框架的支持，尤其感谢下列人员：Ron Akers（Motorola）、Al Aho（Lucent）、Steve Bachinsky（SAIC）、Detlef Becker（Siemens）、Jim Blaine（Washington University）、John Buttitto（Motorola）、Becky Callison（Boeing）、Wei Chiang（Nokia）、Russ Claus（NASA）、Joe Cross（Lockheed Martin）、Bryan Doerr（Boeing）、

Karlheinz Dorn（Siemens）、Sylvester Fernandez（Lockheed Martin）、Andreas Geisler（Siemens）、Helen Gill（DARPA）、Trey Grubbs（Raytheon）、Jody Hagins（ATD）、Andy Harvey（Cisco）、Thomas Heimke（Siemens）、Kalai Kalaichelvan（Nortel）、Arvind Kaushal（Motorola）、Steve Kay（Tellabs）、Chandra Kintala（Lucent）、Gary Koob（DARPA）、Sean Landis（Motorola）、Rick Lett（Sprint）、Joe Loyall（BBN）、Mike Masters（NSWC）、Ed Mays（US Marine Corps）、John Mellby（Raytheon）、Dave Meyer（Virtual Technology）、Eileen Miller（Lucent）、Stan Moyer（Telcordia）、Russ Noseworthy（Object Sciences）、Guru Parulkar（Cisco）、Dan Paulish（Siemens）、James Plamondon（Microsoft）、Dieter Quehl（Siemens）、Lucie Robillard（US Air Force）、Allyn Romanow（Cisco）、Rick Schantz（BBN）、Steve Shaffer（Kodak）、Dave Sharp（Boeing）、Naval Sodha（Ericsson）、Brian Stacey（Nortel）、Paul Stephenson（Ericsson）、Umar Syyid（Hughes）、Dave Thomas（OTI）、Lothar Werzinger（Krones）、Shalini Yajnik（Lucent）以及 Tom Ziomek（Motorola）。

特别感谢文字编辑 Steve Rickaby 对书稿的润色。另外，我们要感谢编辑 Gaynor Redvers-Mutton 以及 John Wiley & Sons 出版社的其他人员，没有他们就没有本书。这是 Gaynor 与 Steve 负责出版的第二本书。他们的支持无与伦比，我们期待未来继续合作出版后续的 POSA 系列书籍。

最后，我们向已故的 Richard Stevens 致以最深切的感谢。多年以前是他的启蒙书籍引导我们走进网络编程的神奇世界，他的精神渗透于本书的字里行间。

阅读指南

“柴郡猫，你能告诉我该走哪条路吗？”

“这要看你想去哪里。”柴郡猫说道。

“我不太在乎去哪里。”爱丽丝说道。

“那你走哪条路都无所谓。”猫咪说道。

“只要我能到达某个地方就行。”爱丽丝补充道。

“哦，那是当然，”柴郡猫说，“只要你走的时间够长。”

——《爱丽丝梦游仙境》，刘易斯·卡罗尔

本书有自己的组织结构，读者可以从头至尾通读本书。但是，如果知道自己想读哪一部分，也可以按照自己的方式来阅读。此时，下面的提示可以帮助读者选择想要关注的主题以及阅读顺序。

模式导论

如果在阅读本书之前没有接触过模式，我们建议读者先阅读[POSA1]和[GoF95]中的模式导论部分，该导论探讨了软件架构和设计方面与模式相关的概念和术语。需要指出的是，本书中的所有模式都建立在[POSA1]介绍的模式概念基础之上，包括：

- 软件架构模式的定义；
- 模式的分类（架构模式、设计模式和成例[①]）；
- 模式的描述格式。

此外，本书中很多模式的实现都使用了[POSA1]及[GoF95]中的模式并加以改良。因此，为了在软件开发项目中使用这些模式，我们建议你把这三本书放在案头，以备查阅。

① 关于这些模式分类的定义参见 8.1 节。

结构与内容

本书第 1 章描述了设计并发与联网系统时面临的主要挑战，然后概述了我们所介绍模式的适用范围与环境，最后展示了一个案例研究：使用本书介绍的 8 个模式开发一个并发的 Web 服务器。

16 个模式描述和一个成例构成了本书的主体。我们将这些模式和成例分四章进行介绍，分别对应并发与联网中间件和应用程序开发过程中面临的关键问题领域：服务访问与配置、事件处理、同步、并发。这些材料的阅读顺序可以自行确定，其中一种阅读顺序是首先阅读重要的核心模式：

- Wrapper Facade设计模式；
- Reactor架构模式；
- Acceptor-Connector设计模式；
- Active Object设计模式。

本书其余 12 个模式与一个成例的安排则尽量避免引用尚未描述的模式。当然读者可以按任意顺序来学习这些模式。这些材料完成并补充了上面列举的四个模式所定义的概念，并涵盖了有效设计和实现并发与联网对象相关的各种问题。

读者也可以使用本书查找解决方案，解决在项目中遇到的问题。可以先使用第 6 章中的模式概览帮助查询，然后再对可能作为解决方案的模式从第 2 章到第 5 章中找到相关的详细描述。

任何模式都不是一座孤岛，不会与其他模式彻底隔绝。因此，第 6 章描述了如何把本书中的所有模式组织起来形成一种模式语言，以服务于构建联网的应用程序和中间件。如果想在深入钻研单个模式之前了解本书模式的全貌，我们推荐读者先走马观花地略读第 6 章介绍的模式语言，然后再深入阅读第 2 章到第 5 章的模式。

第 7 章与第 8 章是本书的最后两章，其中包括术语表、图表中所用符号概述、相关文献的引用以及索引。

模式格式

本书介绍的全部模式均自成一体，沿用了[POSA1]的模式格式。使用该模式格式，既能描述模式的核心内容，又能描述模式的关键细节。我们的目标是：既服务于那些只是想简单地了解模式基本知识的读者，又服务于那些想深入了解模式工作原理的读者。

在本书的模式格式中，每一节均对后续部分做了铺垫。例如，“示例”部分引出了“背景”、“问题”以及“解决方案”，而这几部分则概括了模式的核心内容；“解决方案”部分则为“结构”和“交互”做了铺垫，然后这两部分更为详细地描述了模式的工作原理，这也为读者理解后续的

“实现”做好了铺垫。

接下来的“示例解答”、“变种”、“已知应用”、“效果”以及“参见”部分完成了对每个模式的描述。书中还包含了大量的交叉引用，以便读者理解本书与其他出版物所讲模式之间的关系。

为了将模式实现过程的描述与生产软件系统挂钩，很多示例代码都有 ACE 框架[Sch97]所提供组件的影子。如果只是想先知道所有模式的概况，读者可以在初读本书时跳过“实现”部分，然后当需要知道特定模式的实现细节时再回过头来仔细阅读该部分。

尽管本书使用的模式格式会导致模式的描述略有重复，但是我们发现这种重复可以将“回溯查阅”的可能性降到最低，从而帮助读者更有效地浏览对模式的描述。

在使用图表解释模式的结构与行为时，我们尽量遵循 UML 的标准。但在极少数情况下，UML 并不能足够准确地表达我们的意思，所以正如 8.2 节指出的那样，我们稍微“扩展”了 UML 的标准符号。

背景知识

对于很多模式，尤其是 Reactor、Proactor、Half-Sync/Half-Async 以及 Leader/Followers，我们假定读者熟悉以下内容。

- 面向对象的设计技术，例如模式 [GoF95] [POSA1]与成例[Cope92]、UML符号[BRJ98]以及结构化编程原理，特别是封装与模块化 [Mey97]。
- 面向对象的编程语言特性，例如类[Str97]、继承与多态（polymorhism）[AG98]以及参数化类型[Aus98]。尽管我们介绍了大多数模式的Java已知应用，但是本书的很多示例都是使用C++实现的。
- 系统编程的概念与机制，比如进程与线程管理[Lew95] [Lea99a] [Ric97]、同步 [Ste98]以及进程间通信[Ste99]。
- 网络服务与协议，比如客户端-服务器计算 [CoSte92]以及因特网协议 [Ste93] [SW94]。

本书涉及丰富的专业术语和参考书目，以区分各种技术概念。本书还针对读者可能会进一步学习的主题，提供了相关信息来源。本书毕竟不是一本并发与网络编程的入门教程，所以如果对以上列举的内容不熟悉，我们建议读者结合本书对我们推荐的相关材料做一些背景阅读。

目　录

第 1 章　并发与联网对象 ……1
1.1　驱动因素 ……1
1.2　并发与联网软件面临的挑战 ……4
1.2.1　挑战 1：服务访问与配置 ……6
1.2.2　挑战 2：事件处理 ……9
1.2.3　挑战 3：并发 ……11
1.2.4　挑战 4：同步 ……13
1.2.5　联网软件的其他挑战 ……14
1.3　案例研究：设计一个并发的 Web 服务器程序 ……16
1.3.1　JAWS 框架概况 ……17
1.3.2　使用模式解决 JAWS 中常见的设计挑战 ……18
1.3.3　封装底层的操作系统 API ……19
1.3.4　将事件分离与连接管理同协议处理分离 ……20
1.3.5　通过多线程按比例提升服务器程序的性能 ……21
1.3.6　实现同步请求队列 ……23
1.3.7　最小化服务器程序的线程开销 ……24
1.3.8　有效利用异步 I/O ……25
1.3.9　增强服务器程序的可配置性 ……27
1.3.10　其他用于实现 JAWS 的模式 ……28
1.4　小结 ……29

第 2 章　服务访问和配置模式 ……31
2.1　Wrapper Facade ……32
2.2　Component Configurator ……51
2.3　Interceptor ……73
2.4　Extension Interface ……95

第 3 章　事件处理模式 ……119
3.1　Reactor ……120
3.2　Proactor ……146
3.3　Asynchronous Completion Token ……178
3.4　Acceptor-Connector ……195

第 4 章　同步模式 ……223
4.1　Scoped Locking ……224
4.2　Strategized Locking ……230
4.3　Thread-Safe Interface ……238
4.4　Double-Checked Locking Optimization ……244

第 5 章　并发模式 ……253
5.1　Active Object 模式 ……254
5.2　Monitor Object 模式 ……275
5.3　Half-Sync/Half-Async 模式 ……292
5.4　Leader/Followers 模式 ……306
5.5　Thread-Specific Storage 模式 ……324

第 6 章　模式的综合运用 ……345
6.1　从单个模式到模式语言 ……345
6.1.1　没有模式是一座孤岛 ……345
6.1.2　模式语言的形成 ……347
6.2　面向中间件和应用程序的模式语言 ……348
6.2.1　模式语言的细节 ……348
6.2.2　模式语言之我见 ……355
6.3　并发与联网之余 ……356
6.3.1　图形用户接口 ……356
6.3.2　组件 ……357
6.3.3　通用编程 ……357
6.4　模式语言与模式系统 ……358