

Patterns of Enterprise Application Architecture

# 企业应用架构模式

## (英文版)

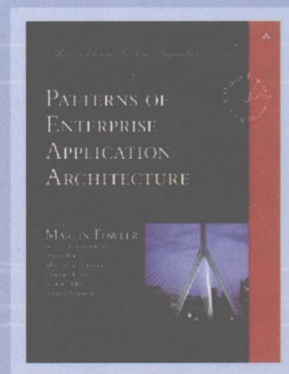
[美] Martin Fowler 著

2003年Jolt生产效率奖得主

2003年Software Development杂志读者选择大奖得主

2003年JavaWorld.com编辑选择奖得主

- 企业应用开发圣经
- 软件开发大师代表作，兼顾各种主流开发平台
- 传授高手之间口口相传的“内功心法”

人民邮电出版社  
POSTS & TELECOM PRESS



“本书是企业级应用设计与开发领域的杰作，处处闪烁着思想的光芒，发人深省。”

——Jolt奖颁奖词

# Patterns of Enterprise Application Architecture

## 企业应用架构模式 (英文版)

随着信息技术的广泛应用，系统需要处理的数据量越来越大，企业级软件开发已经渐成主流，而开发人员面临的困难与挑战也是显而易见的。更糟糕的是，这一领域的资料一直非常缺乏。

本书是软件开发大师Martin Fowler的代表作，采用模式的形式系统总结了业界多年积累的经验，被称为“企业级应用开发领域的圣经”，出版以来一直畅销不衰，至今仍然无可替代。作者在精彩地阐述了企业应用开发和设计中的核心原则基础上，详细、生动地讲述了51个模式并给出主流平台（Java和.NET）中的应用实例，更分析了许多相似模式之间的差异，提供了具体运用和选择这些模式的大量经验之谈，使你不仅知其然，更知其所以然。

这是一部软件开发领域不朽的经典，任何一位真正的软件开发人员都不可错过。



**Martin Fowler** 享誉世界的软件开发大师，现为著名软件开发咨询公司ThoughtWorks的首席科学家。他在面向对象分析与设计、UML、设计模式、软件开发方法学、XP、重构等方面都有重要贡献。他更是全球最具影响力的技术作家之一，除本书外，他的《分析模式》、《UML精粹》和《重构》等著作都已经成为经典。

### 延伸阅读

重构：改善既有代码的设计（英文注释版） **Martin Fowler** 著  
敏捷软件开发：原则、模式与实践（英文注释版） **Robert C. Martin** 著  
程序员修炼之道（英文注释版） **Andrew Hunt David Thomas** 著  
编程精粹：编写高质量C语言代码（英文版） **Steve Maguire** 著  
我们在微软怎样开发软件（英文版） **Donis Marshall John Bruno** 著  
敏捷软件开发：原则、模式与实践（C#版） **Robert C. Martin Micah Martin** 著  
修改代码的艺术 **Michael Feathers** 著  
高质量程序设计艺术 **Diomidis Spinellis** 著



www.pearsonhighered.com

本书相关信息请访问：图灵网站 <http://www.turingbook.com>

读者/作者热线：(010)51095186

反馈/投稿/推荐信箱：contact@turingbook.com

**分类建议** 计算机/程序设计

人民邮电出版社网址：www.ptpress.com.cn



ISBN 978-7-115-20422-6



9 787115 204226 >

ISBN 978-7-115-20422-6/TP

定价：79.00元

TURING

图灵程序设计丛书

程序员修炼系列

Patterns of Enterprise Application Architecture

# 企业应用架构模式

(英文版)

[美] Martin Fowler 著

人民邮电出版社  
北京

## 图书在版编目 (CIP) 数据

企业应用架构模式 = Patterns of Enterprise Application Architecture: 英文 / (美) 福勒 (Fowler, M.)

著. —北京: 人民邮电出版社, 2009.6

(图灵程序设计丛书)

ISBN 978-7-115-20422-6

I. 企… II. 福… III. 计算机应用—企业管理—英文  
IV. F270.7

中国版本图书馆CIP数据核字 (2009) 第020114号

## 内 容 提 要

面向对象大师 Martin Fowler 及其专家级合作者将 40 多种常用解决方案转化成模式, 为我们提供了这本能够应用于任何一种企业应用平台的、关于解决方案的参考书。本书叙述深入浅出, 采用大量 UML 图进一步阐明有关概念。前面介绍企业应用的背景知识, 如分层架构、Web 表现、业务逻辑、数据库映射、并发、会话、分布策略等。在此基础上, 随后的各章分别对与这些背景知识相关的设计模式进行了详细的介绍, 并配以详细的 Java 代码或 C# 代码示例。

本书适合设计和构建企业应用的软件架构师、设计人员和编程人员阅读, 同时也可作为高等院校计算机专业及软件学院相关课程的参考教材。

图灵程序设计丛书

## 企业应用架构模式 (英文版)

◆ 著 [美] Martin Fowler

责任编辑 傅志红

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街14号

邮编 100061 电子函件 315@ptpress.com.cn

网址 <http://www.ptpress.com.cn>

北京铭成印刷有限公司印刷

◆ 开本: 800×1000 1/16

印张: 34.75

字数: 667千字

印数: 1—3 000册

2009年6月第1版

2009年6月北京第1次印刷

著作权合同登记号 图字: 01-2006-3693号

ISBN 978-7-115-20422-6/TP

定价: 79.00元

读者服务热线: (010)51095186 印装质量热线: (010)67129223

反盗版热线: (010)67171154



# 版 权 声 明

Original edition, entitled *Patterns of Enterprise Application Architecture*, 0321127420 by Martin Fowler, published by Pearson Education, Inc., publishing as Addison-Wesley, Copyright © 2003 by Pearson Education, Inc.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc.

China edition published by PEARSON EDUCATION ASIA LTD and POSTS & TELECOM PRESS Copyright © 2009.

This edition is manufactured in the People's Republic of China, and is authorized for sale only in the People's Republic of China excluding Hong Kong, Macao and Taiwan.

本书英文版由 Pearson Education Asia Ltd.授权人民邮电出版社独家出版。未经出版者书面许可，不得以任何方式复制或抄袭本书内容。

仅限于中华人民共和国境内（香港、澳门特别行政区和台湾地区除外）销售发行。

本书封面贴有 Pearson Education（培生教育出版集团）激光防伪标签，无标签者不得销售。

版权所有，侵权必究。

*For Denys William Fowler, 1922–2000  
in memoriam*

*—Martin*

# 前言

1999年春天，我飞抵芝加哥为ThoughtWorks公司正在开发的一个项目担任顾问。ThoughtWorks是一个规模虽小但正在快速成长的应用开发公司。这个项目属于那种雄心勃勃的企业级应用——一个后端租赁系统。简单说，它处理的是租户签字认可后所有与租赁有关的事务，包括发送账单、处理某些人对所租房产的改造、追踪那些未按时付款的租户、处理提前退租的情况等。这听起来好像并不太难实现，但是你大概没有意识到，租赁合同极度复杂并且总在不断变化。它的业务“逻辑”几乎不能套用任何已有的逻辑模式，因为那些“逻辑”归根到底是商人们为了抢生意而制定的，一些奇怪的小改动都可能对赢得某笔生意起关键作用。而每次这样的成功就意味着系统复杂性的增加。

我感兴趣的正是这类问题：如何捕获这些复杂情况，并设计一个面向对象的系统使问题更容易解决。事实上，我认为面向对象的最大优点就在于它能够使复杂逻辑易于处理。为复杂业务问题开发一个好的领域模型（Domain Model, 116）很困难，但结果往往非常令人满意。

但问题还没有完。我们的领域模型必须持久存储到数据库中，与许多项目一样，我们使用的也是关系数据库。我们还必须将该模型与用户界面连接起来，还要支持远程应用程序使用本软件，并将软件与第三方的软件包集成。所有这些工作都基于一种称作J2EE的新技术，当时世界上没有人在这方面具备真正的实战经验。

虽然J2EE当时还很新，但我们可以受助于以往的经验。我曾经使用C++、Smalltalk和CORBA开发类似系统已经多年。ThoughtWorks公司中许多人在Forte<sup>①</sup>方面有丰富经验。我们已经有了关键架构的思路，剩下的只是弄清如何用J2EE实现。三年后再来回顾，虽然设计并不完美，但却很好地经受住了时间的考验。

本书正是为这样的情形而撰写的。多年来，我见证了许多企业级应用项目。这些项目通常都包含相似的设计思路，事实证明这些设计思路可以有效地处理企业应用中不可避免的复杂性。本书就是将这些设计思路以模式形式表述的一个起点。

本书分为两个部分，第一部分是一些叙述性的章节，讨论企业级应用程序

---

① Forte 是一个专有的应用服务器，其开发公司后被 Sun 收购，产品分别并入不同产品线。

设计中的一些重要主题。这些章节介绍了企业级应用程序架构的各种问题及其解决方案，但解决方案的细节没有深入讨论。解决方案的细节是在本书的第二部分中以模式的方式组织起来的。这些模式都是参考，我并不希望读者逐页阅读。我的意图是，读者从头到尾将第一部分的叙述性章节读完，然后再根据兴趣和需求浏览第二部分的模式章节。因此，本书实际上是一本简短的叙述图书和详尽的参考书的二合一。

本书讨论的是企业级应用程序的设计。企业级应用程序涉及大量复杂数据的显示、操作和存储以及对这些数据进行处理的业务流程的自动化。典型的例子有预订系统、财务系统、供应链系统以及许多其他支持现代商业运作的系统。企业级应用和嵌入式系统、控制系统、电信系统或者桌面应用软件不同，它们有自己特有的难度和解决方案。因此，如果你从事的是上述那些非企业领域的工作，那么本书不适合你（除非想体会一下企业级应用程序是怎么一回事）。如果需要一本关于软件架构的通用书，我推荐[POSA]。

在构建企业级应用时有许多架构方面的问题。本书恐怕无法面面俱到。在软件开发方面，我一直坚信迭代开发方法。迭代开发的核心在于只要软件具备了用户可用的功能，就应当交付，即使软件并未完成。虽然著书与编写软件之间存在很多差异，但在这一点上我认为是相通的。也就是说，虽然本书尚不全面，但（我相信）可以为读者提供有关企业级应用程序架构方面的有益建议。本书讨论的主题包括：

- 企业级应用程序的分层；
- 构建领域（业务）逻辑；
- 构建 Web 用户界面；
- 将内存中的模块（特别是对象）与关系数据库关联起来；
- 处理在无状态环境下的会话状态；
- 分布的原则。

本书尚未涉及的主题更多。我很想撰写关于验证的组织、消息和异步通信的加入、安全、错误处理、集群、应用集成、架构重构、构建富客户用户界面等方面的书籍或文章。但是，由于时间和篇幅的限制以及想法尚不成熟，本书没有涉及上述内容。我希望在不久的将来能看到这方面的模式。也许我会撰写另一本书讨论这些内容，或者会有其他人来弥补这一缺憾。

其中基于消息的通信是相当重要的问题。从事多应用程序集成的开发者正越来越多地用到异步的、基于消息的通信方法。在应用程序内部的应用也有很多内容需要探讨。

本书并不针对特定的软件平台。从 20 世纪 80 年代末到 90 年代初，我开始在 Smalltalk、C++ 和 CORBA 项目中使用这些模式。而 20 世纪 90 年代后期我



开始在 Java 方面做大量的工作，我发现这些模式可以很好地应用于较早的 Java/CORBA 系统和其后基于 J2EE 的开发。最近，我开始探索微软的 .NET 平台，发现这些模式同样有效。ThoughtWorks 公司的同事也介绍了他们的经验，尤其是 Forte 方面。我不敢说这些模式能够适用于所有已经和即将用于企业级应用的开发平台，但到目前为止，它们显示出足够的适用性。

本书为大多数模式提供了代码示例。选择程序设计语言时主要考虑大多数读者都能够阅读和理解。Java 就是一个很好的选择。任何人只要能阅读 C 或 C++ 代码就可以读懂 Java 代码，Java 比 C++ 简单得多。大多数 C++ 程序员都能够理解 Java，但反过来却并非如此。我偏爱面向对象，当然倾向于使用面向对象语言。因此，大多数代码示例使用的是 Java 语言。写作本书时，微软的 .NET 环境逐渐稳定，它的 C# 语言与 Java 有许多相同之处。所以某些代码示例也使用了 C# 语言，虽然这样做会多少有一些风险，因为大多数开发者没有太多 .NET 经验，所以所用的惯用法并不成熟。这两种语言都基于 C，因此只要能阅读其中一种，就算对另一种语言或平台没有深入研究，也能阅读。我的目的是使用一种能够让最多的软件开发人员阅读的语言，虽然这并非他们主要使用或爱用的语言。（谨向那些喜欢 Smalltalk、Delphi、Visual Basic、Perl、Python、Ruby、COBOL 或其他语言的读者致歉。我知道你们认为有比 Java 或 C# 更好的语言，我只能说我也认为如此！）

示例是用来阐述和解释模式思想的。它们并非可以直接使用的解决方案，任何情况下还需要做不少工作才能应用它们。模式只是有益的起点，不是终点。

## 本书面向的读者

本书面向的是正在构建企业级应用，希望增进对架构相关问题的理解，加强这方面沟通的程序员、设计师和软件架构师。

我假定本书的大多数读者可以归为两类：一些人所面临的需求规模不大，计划自己构建软件；另一些人面临的需求较多，需要使用某些工具。对于较小规模的需求，本书提及的模式将有助于着手工作。尽管在许多领域，所需的知识超出了本书的内容，但本书能够提供先发优势。对于工具使用者，我希望本书能揭示工具背后的一些技术内幕，帮助他们决策如何选择工具所支持的模式。例如，使用对象-关系映射工具时，你仍然必须决定如何映射某些特定情况。本书所提供的模式将有助于做出这样的决策。

此外，还存在第三类读者，他们有较多需求，又要自己构建软件。在此，我要说的是：请先考虑是否使用工具。我已经见过不止一个项目花费大量时间来建造框架，而这些框架并非该项目所真正要解决的问题。当然，如果确认没有问题，那请继续好了。请记住，为提高可理解性，本书中许多代码示例都有

意简化了，你会发现需要做很多工作来处理所面临的更多需求。

由于模式是对不断出现的问题的通用解决方案，因此很可能有的读者已经遇到过当中的一些模式了。如果你从事企业级应用开发已经有一段时间，可能会很熟悉其中大部分模式。本书中并不包含任何新的东西，正相反，这是一本讲述（我们这一行业的）既有知识的书。如果你是这一领域的新手，我希望本书将帮助你学习这些技术。如果你熟悉这些技术，我希望本书有助于你与其他人沟通。模式的重要作用就在于其创建了一个通用的词汇表，例如，你称某个类是远程外观（Remote Facade,388），其他设计人员就都知道你指的是什么。

## 致谢

与其他书籍一样，这里将涉及多年来以不同方式与我一起工作的很多人。他们以许多方式对我提供过帮助。本书中一些重要内容是由他人提供的，其中某些人我可能已经无法回忆起他们的名字。在这里，我所能做的是对那些仍铭记在心的人表示感谢。

首先要感谢的是我的合作者，David Rice，他是我在 ThoughtWorks 公司的同事，他为本书做出了巨大的贡献：撰写了本书的 1/10 以上的内容。当我们一起努力以保证本书能如期交稿时（他当时还要从事客户支持的工作），我们曾在若干个深夜通过即时消息软件进行协商，交谈中他坦承他总算明白了为何写一本书是如此困难却又如此吸引人。

Matt Foemmel 是我在 ThoughtWorks 公司的另一个同事。他为代码示例做出了很大的贡献。Randy Stafford 为本书贡献了服务层（Service Layer,133），他一直是此模式的极力倡导者。我还要感谢 Edward Hieatt 和 Rob Mee 所做的贡献，Rob 在审阅本书时发现了一个漏洞。Rob 是我的最佳审稿人：他不仅发现少了某些内容，而且他还帮我写了一节来弥补这个漏洞！

同样，对本书一流的正式审稿人，我的言辞远不能表达我的感激之情：

John Brewer、Rob Mee、Kyle Brown、Gerard Meszaros、Jens Coldewey、Dirk Riehle、John Crupi、Randy Stafford、Leonard Fenster、David Siegel、Alan Knight、Kai Yu。

我几乎要把 ThoughtWorks 公司的电话号码本列全了，因为有太多的同事与我讨论他们的设计和实验，助我完成这一项目。许多模式在我脑中成型，是因为我有机会与众多天才设计师讨论，因此我只好对整个公司表示感谢。

Kyle Brown、Rachel Reintz 和 Bobby Woolf 在百忙之中抽出时间与我一道在北卡罗莱纳对本书进行了长期而细致的审阅。他们的批评指正为本书注入了各种睿智之见。尤其是与 Kyle 的几次长时间的电话交谈令我获益匪浅。

2000 年初我与 Alan Knight 和 Kai Yu 一起为 Java One 大会准备了一个演讲，

这是本书的雏形。我还要感谢 Josh Mackenzie、Rebecca Parsons 和 Dave Rice，他们其后协助我提炼了这些演讲及其思想。Jim Newkirk 付出了很大努力来帮助我熟悉.NET 平台。

我与这个领域的许多专家有过令人愉快的交谈或合作，我从他们那里获益匪浅。这里尤其要对 Colleen Roe、David Muirhead 和 Randy Stafford 表示感谢，他们将自己在 Gemstone 的 Foodsmart 示例系统上的工作成果与我共享。我参加了 Bruce Eckel 所主持的 Crested Butte 讨论会的一些重要会谈，向近年来这一会议的与会者致谢。Joshua Kerievsky 虽然没有时间对本书做一次全面的审阅，但他是我在模式方面很好的顾问。

我一如既往地伊利诺斯大学阅读组独特的不闭门评审会得到了极大的帮助。我要感谢：Ariel Gertzenstein、Bosko Zivaljevic、Brad Jones、Brian Foote、Brian Marick、Federico Balaguer、Joseph Yoder、John Brant、Mike Hewner、Ralph Johnson 和 Weerasak Witthawaskul。

伊利诺斯大学的毕业生 Dragos Manolescu 及其小组给我提供了不少反馈意见。感谢 Muhammad Anan、Brian Doyle、Emad Ghosheh、Glenn Graessle、Daniel Hein、Prabhakaran Kumarakulasingam、Joe Quint、John Reinke、Kevin Reynolds、Sripriya Srinivasan 和 Tirumala Vaddiraju。

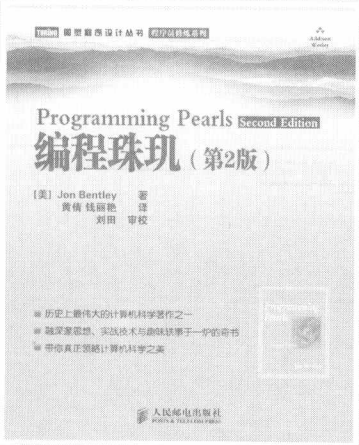
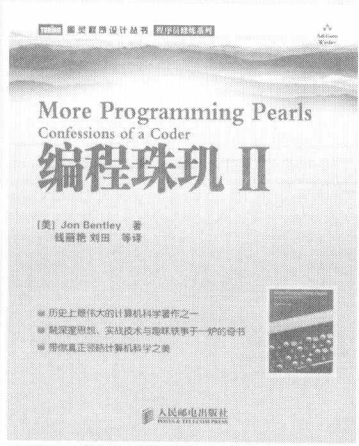
Kent Back 为我提供了许多思路。是他为特殊情况（Special Case,496）模式起了名字。Jim Odell 将我领入了咨询、教学和写作的世界，我由衷地感谢他。

当我写这本书时，曾将草稿放在网上。期间许多人通过电子邮件向我指出问题、提出疑问或者讨论其他替代方案。他们中有 Michael Banks、Mark Bernstein、Graham Berrisford、Bjorn Beskow、Bryan Boreham、Sean Broadley、Peris Brodsky、Paul Campbell、Chester Chen、John Coakley、Bob Corrick、Pascal Costanza、Andy Czerwonka、Martin Diehl、Daniel Drasin、Juan Gomez Duaso、Don Dwiggins、Peter Foreman、Russell Freeman、Peter Gassmann、Jason Gorman、Dan Green、Lars Gregori、Rick Hansen、Tobin Harris、Russel Healey、Christian Heller、Richard Henderson、Kyle Hermenean、Carsten Heyl、Akira Hirasawa、Eric Kaun、Kirk Knoernschild、Jesper Ladegaard、Chris Lopez、Paolo Marino、Jeremy Miller、Ivan Mitrovic、Thomas Neumann、Judy Obee、Paolo Parovel、Trevor Pinkney、Tomas Restrepo、Joel Rieder、Matthew Roberts、Stefan Roock、Ken Rosha、Andy Schneider、Alexandre Semenov、Stan Silvert、Geoff Soutter、Volker Termath、Christopher Thames、Volker Turau、Knut Wannheden、Marc Wallace、Stefan Wenig、Brad Wiemerslage、Mark Windholtz、Michael Yoon。

此外，还有许多我不认识或已经遗忘了的人，在此致以真诚的谢意。

最诚挚的谢意依旧要献给我的妻子 Cindy，感谢她一直陪伴左右。

# 一部20余年畅销不衰的不朽经典



## 本书赞誉

“《编程珠玑》第1版是对我职业生涯早期影响最大的书之一，其中的许多真知灼见多年之后仍然使我受益匪浅。Jon在第2版中对素材进行了大量更新，许多新内容让我耳目一新。”

——Steve McConnell，软件工程大师，  
IEEE Software 前主编，《代码大全》作者

“对每一位遇到的程序员，我都会毫不迟疑地建议他阅读并不断重读这部经典之作。”

——Slashdot

## 内容简介

多年以来，当程序员们推选出最心爱的计算机图书时，《编程珠玑》总是位于前列。正如自然界里珍珠出自细沙对牡蛎的磨砺，计算机科学大师 Jon Bentley 以其独有的洞察力和创造力，从磨砺程序员的实际问题中凝结出一篇篇不朽的编程“珠玑”，成为世界计算机界名刊《ACM 通讯》历史上最受欢迎的专栏，最终结集为两部不朽的计算机科学经典名著，影响和激励着一代又一代程序员和计算机科学工作者。

## 作者简介

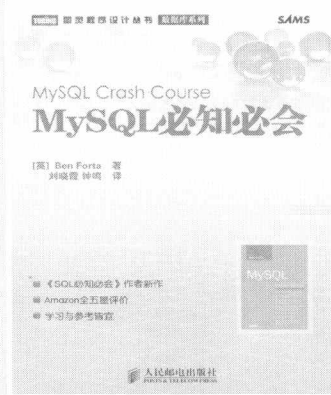
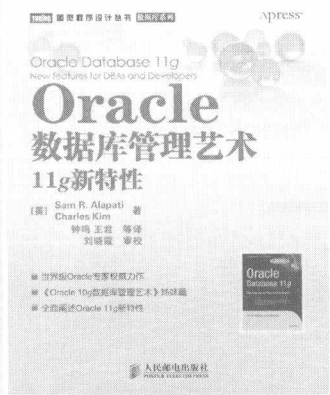
Jon Bentley 世界著名计算机科学家，被誉为影响算法发展的十位大师之一。他先后任职于卡内基-梅隆大学（1976~1982）、贝尔实验室（1982~2001）和 Avaya 实验室（2001 年至今）。在卡内基-梅隆大学担任教授期间，他培养了包括 Tcl 语言设计者 John Ousterhout、Java 语言设计者 James Gosling、《算法导论》作者之一 Charles Leiserson 在内的许多计算机科学大家。2004 年荣获 Dr. Dobb's 程序设计卓越奖。

- 历史上最伟大的计算机科学著作之一
- 融深邃思想、实战技术与趣味轶事于一炉的奇书
- 带你真正领略计算机科学之美



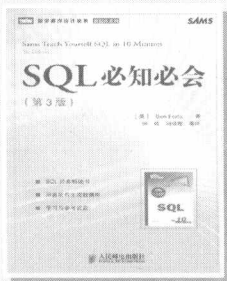
# 数据库系列

○ 世界级Oracle专家权威力作  
○ 《Oracle 10g数据库管理艺术》姊妹篇  
○ 全面阐述Oracle 11g新特性

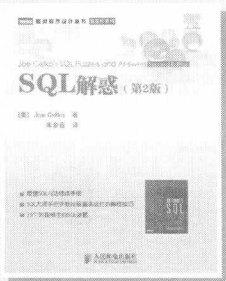


○ 《SQL必知必会》作者新作  
○ Amazon全五星评价  
○ 学习与参考皆宜

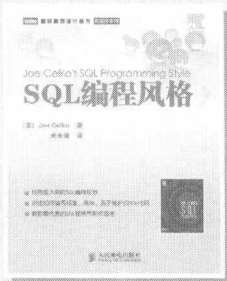
## 图灵数据库经典系列



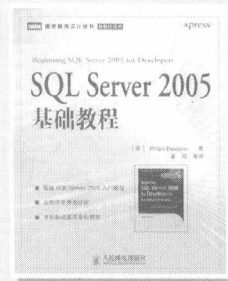
○ SQL经典畅销书  
○ 涵盖所有主流数据库  
○ 学习与参考皆宜



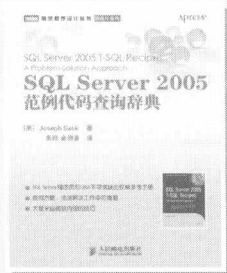
○ 最佳SQL内功修炼手册  
○ SQL大师手把手教授极富实战性的编程技巧  
○ 75个妙趣横生的SQL谜题



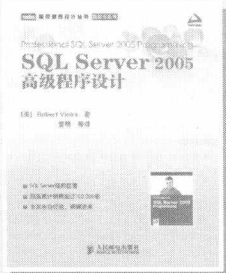
○ 世界级大师的SQL编程规范  
○ 讲述如何编写标准、高效、易于维护的SQL代码  
○ 教你像优秀的SQL程序员那样思考



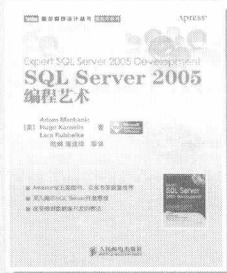
○ 最佳SQL Server 2005入门教程  
○ 采用中文界面讲述  
○ 兼顾数据库开发和管理



○ SQL Server程序员和DBA不可或缺权威参考手册  
○ 查询方便，迅速解决工作中的难题  
○ 大量来自微软内部的技巧



○ SQL Server经典巨著  
○ 原版累计销售超过100000册  
○ 专家亲身经验，娓娓道来



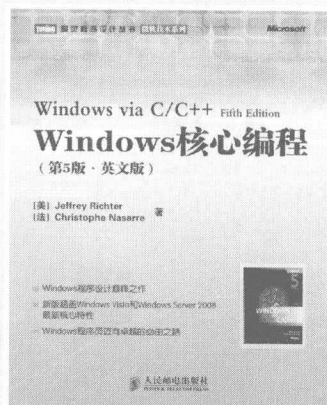
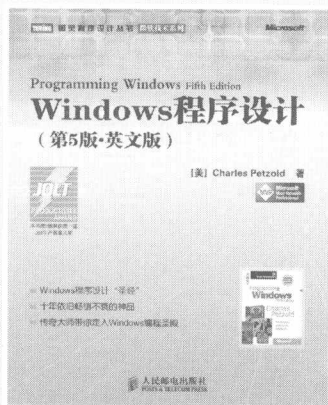
○ Amazon全五星图书，众多专家盛誉推荐  
○ 深入揭示SQL Server开发思想  
○ 改变你对数据库开发的看法



○ 《SQL必知必会》作者新作  
○ Amazon全五星评价  
○ 学习与参考皆宜

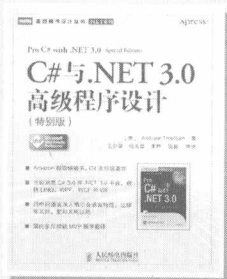
# 微软技术系列

Windows 程序设计“圣经”  
十年依旧畅销不衰的神品  
传奇大师带你走入 Windows 编程圣殿

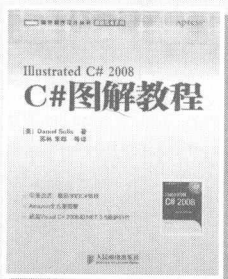


Windows 程序设计巅峰之作  
新版涵盖 Windows Vista 和 Windows Server 2008 最新核心特性  
Windows 程序员迈向卓越的必由之路

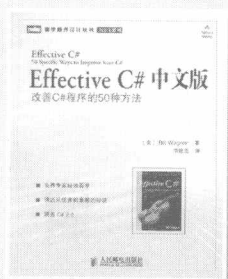
## 图灵C#/.NET经典·畅销系列



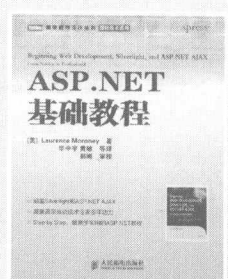
Amazon 超级畅销书，C# 圣级著作  
全面涵盖 C# 3.0 和 .NET 3.0 平台，包括 LINQ、WPF、WCF 和 WF  
用中间语言深入揭示各语言特性，让你知其然，更知其所以然



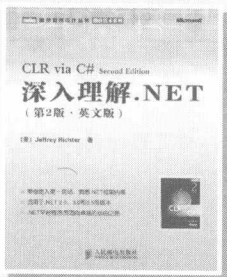
用图说话，最易学的 C# 教程  
Amazon 全五星盛誉  
涵盖 Visual C# 2008 和 .NET 3.5 最新特性



业界专家经验荟萃  
讲述从优秀到卓越的秘诀  
涵盖 C# 2.0



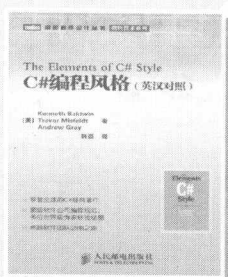
涵盖 Silverlight 和 ASP.NET AJAX  
凝聚资深微软技术专家多年功力  
Step by Step，最易学实用的 ASP.NET 教程



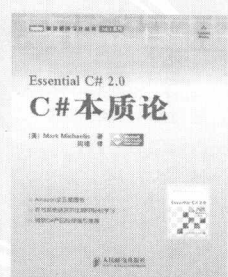
带你走入第一现场，洞悉 .NET 框架内幕  
适用于 .NET 2.0、3.0 和 3.5 各版本  
.NET 平台程序员迈向卓越的必由之路



来自微软 .NET 框架设计组的智慧结晶  
洞悉 .NET 技术内幕  
.NET 开发者的必备图书



享誉全球的 C# 经典著作  
顶级软件公司编程规范，多位世界级专家经验结晶  
卓越软件团队必由之路



Amazon 全五星图书  
在与其他语言的比较中轻松学习  
微软 C# 产品经理强烈推荐

# Contents

|                                                    |           |
|----------------------------------------------------|-----------|
| <b>Introduction</b>                                | <b>1</b>  |
| Architecture                                       | 1         |
| Enterprise Applications                            | 2         |
| Kinds of Enterprise Application                    | 5         |
| Thinking About Performance                         | 6         |
| Patterns                                           | 9         |
| The Structure of the Patterns                      | 11        |
| Limitations of These Patterns                      | 13        |
| <b>PART 1: The Narratives</b>                      | <b>15</b> |
| <b>Chapter 1: Layering</b>                         | <b>17</b> |
| The Evolution of Layers in Enterprise Applications | 18        |
| The Three Principal Layers                         | 19        |
| Choosing Where to Run Your Layers                  | 22        |
| <b>Chapter 2: Organizing Domain Logic</b>          | <b>25</b> |
| Making a Choice                                    | 29        |
| Service Layer                                      | 30        |
| <b>Chapter 3: Mapping to Relational Databases</b>  | <b>33</b> |
| Architectural Patterns                             | 33        |
| The Behavioral Problem                             | 38        |

|                                                                             |           |
|-----------------------------------------------------------------------------|-----------|
| Reading in Data .....                                                       | 40        |
| Structural Mapping Patterns .....                                           | 41        |
| Mapping Relationships .....                                                 | 41        |
| Inheritance .....                                                           | 45        |
| Building the Mapping .....                                                  | 47        |
| Double Mapping .....                                                        | 48        |
| Using Metadata .....                                                        | 49        |
| Database Connections .....                                                  | 50        |
| Some Miscellaneous Points .....                                             | 52        |
| Further Reading .....                                                       | 53        |
| <b>Chapter 4: Web Presentation .....</b>                                    | <b>55</b> |
| View Patterns .....                                                         | 58        |
| Input Controller Patterns .....                                             | 61        |
| Further Reading .....                                                       | 61        |
| <b>Chapter 5: Concurrency (<i>by Martin Fowler and David Rice</i>).....</b> | <b>63</b> |
| Concurrency Problems .....                                                  | 64        |
| Execution Contexts .....                                                    | 65        |
| Isolation and Immutability .....                                            | 66        |
| Optimistic and Pessimistic Concurrency Control .....                        | 67        |
| Preventing Inconsistent Reads .....                                         | 68        |
| Deadlocks .....                                                             | 70        |
| Transactions .....                                                          | 71        |
| ACID .....                                                                  | 71        |
| Transactional Resources .....                                               | 72        |
| Reducing Transaction Isolation for Liveness .....                           | 73        |
| Business and System Transactions .....                                      | 74        |
| Patterns for Offline Concurrency Control .....                              | 76        |
| Application Server Concurrency .....                                        | 78        |
| Further Reading .....                                                       | 80        |
| <b>Chapter 6: Session State .....</b>                                       | <b>81</b> |
| The Value of Statelessness .....                                            | 81        |
| Session State .....                                                         | 83        |
| Ways to Store Session State .....                                           | 84        |
| <b>Chapter 7: Distribution Strategies .....</b>                             | <b>87</b> |
| The Allure of Distributed Objects .....                                     | 87        |
| Remote and Local Interfaces .....                                           | 88        |

|                                                            |            |
|------------------------------------------------------------|------------|
| Where You Have to Distribute .....                         | 90         |
| Working with the Distribution Boundary .....               | 91         |
| Interfaces for Distribution .....                          | 92         |
| <b>Chapter 8: Putting It All Together .....</b>            | <b>95</b>  |
| Starting with the Domain Layer .....                       | 96         |
| Down to the Data Source Layer .....                        | 97         |
| Data Source for <i>Transaction Script</i> (110) .....      | 97         |
| Data Source for <i>Table Module</i> (125) .....            | 98         |
| Data Source for <i>Domain Model</i> (116) .....            | 98         |
| The Presentation Layer .....                               | 99         |
| Some Technology-Specific Advice .....                      | 100        |
| Java and J2EE .....                                        | 100        |
| .NET .....                                                 | 101        |
| Stored Procedures .....                                    | 102        |
| Web Services .....                                         | 103        |
| Other Layering Schemes .....                               | 103        |
| <b>PART 2: The Patterns .....</b>                          | <b>107</b> |
| <b>Chapter 9: Domain Logic Patterns .....</b>              | <b>109</b> |
| Transaction Script .....                                   | 110        |
| How It Works .....                                         | 110        |
| When to Use It .....                                       | 111        |
| The Revenue Recognition Problem .....                      | 112        |
| Example: Revenue Recognition (Java) .....                  | 113        |
| Domain Model .....                                         | 116        |
| How It Works .....                                         | 116        |
| When to Use It .....                                       | 119        |
| Further Reading .....                                      | 119        |
| Example: Revenue Recognition (Java) .....                  | 120        |
| Table Module .....                                         | 125        |
| How It Works .....                                         | 126        |
| When to Use It .....                                       | 128        |
| Example: Revenue Recognition with a Table Module (C#) .... | 129        |
| Service Layer ( <i>by Randy Stafford</i> ) .....           | 133        |
| How It Works .....                                         | 134        |
| When to Use It .....                                       | 137        |