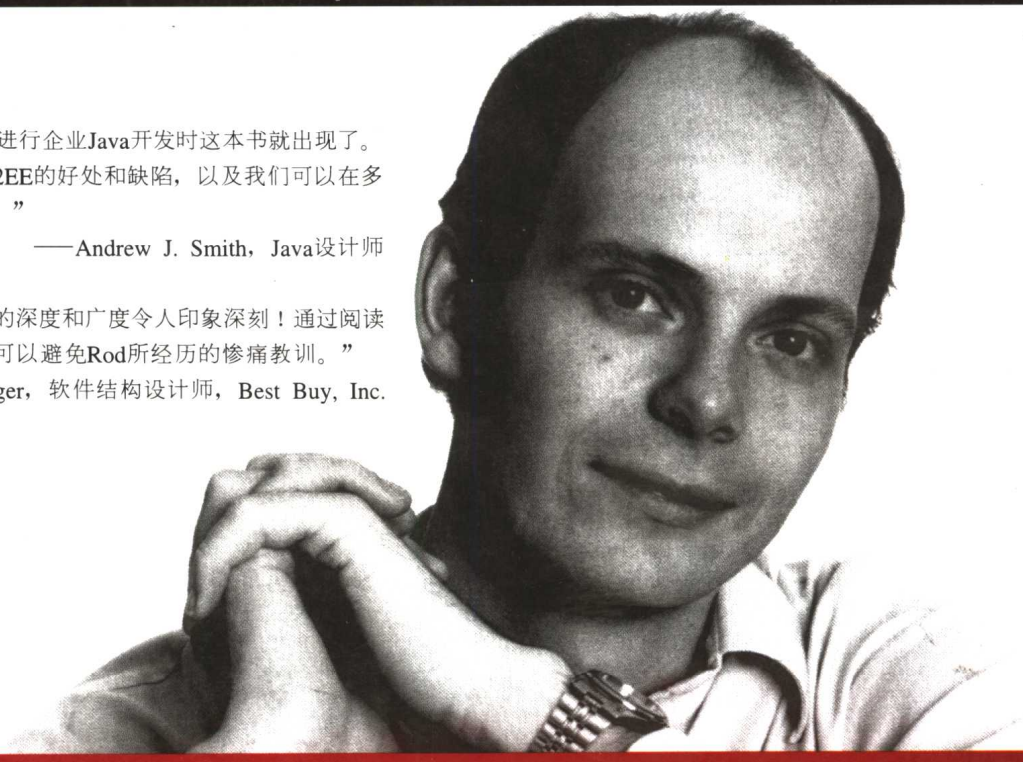
Expert one-on-one J2EE Design and Development



“真希望在我刚开始进行企业Java开发时这本书就出现了。这本书告诉了我们使用J2EE的好处和缺陷，以及我们可以在多大程度上避免这些缺陷。”

——Andrew J. Smith，Java设计师

“Rod所拥有的经验的深度和广度令人印象深刻！通过阅读这本书，J2EE开发人员可以避免Rod所经历的惨痛教训。”

——Todd Lauinger，软件结构设计师，Best Buy, Inc.

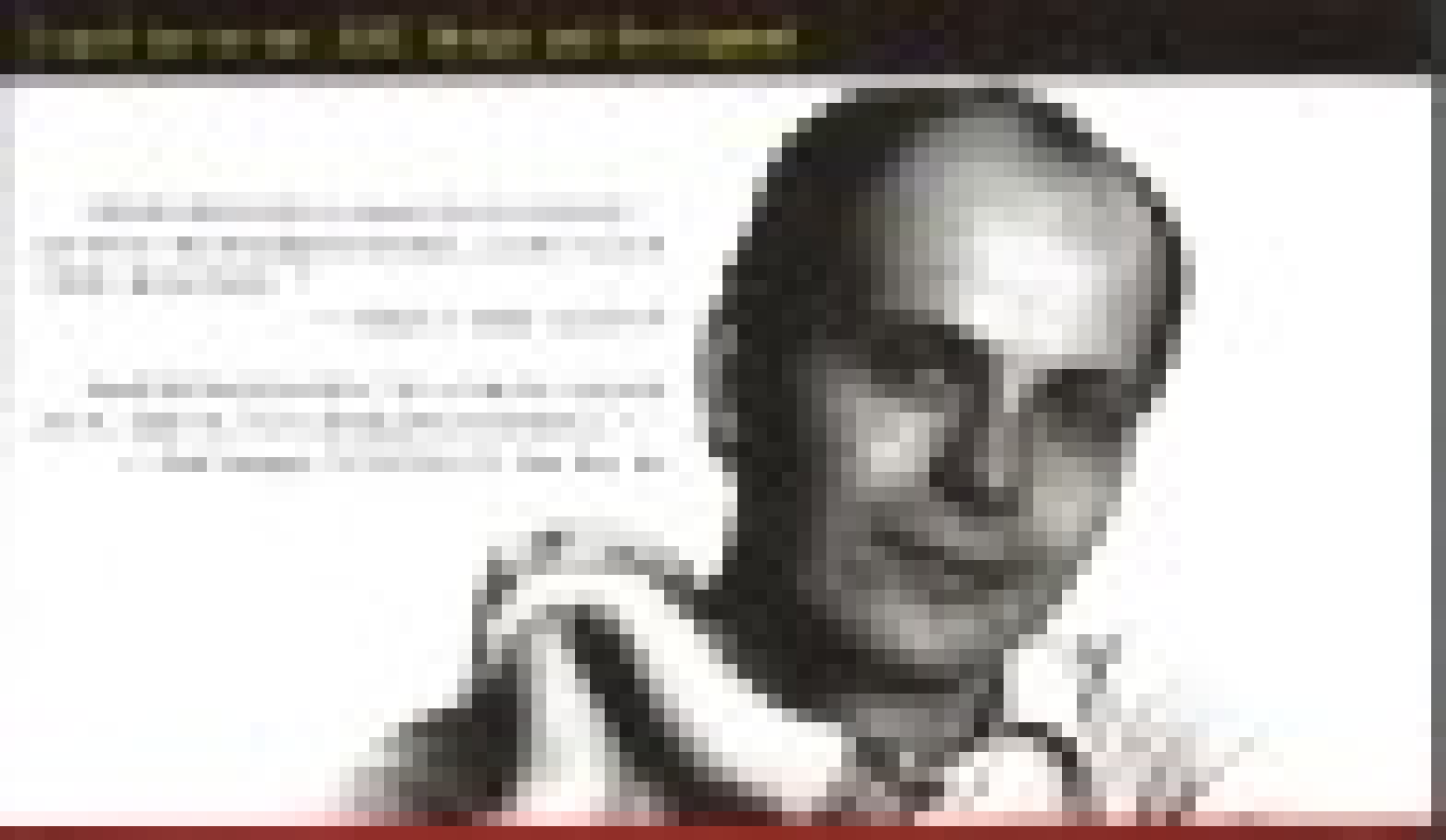
J2EE
设计开发编程指南

〔美〕 Rod Johnson 著

魏海萍 于晓菲 毛 选 等译

wrox
PROGRAMMER TO PROGRAMMER™

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



J2EE
设计开发编程指南

Expert one-on-one J2EE Design and D

J2EE设计开发编程指南

〔美〕 Rod Johnson 著

魏海萍 于晓菲 毛 选 等译

電子工業出版社

Publishing House of Electronics Industry

北京 •

内 容 提 要

J2EE是当今可用于企业软件开发的最佳平台。本书的目标是让读者能够轻松自如地制定J2EE开发的体系结构决策与实现决策。内容涉及：在何种情况下使用分布式体系结构；如何高效地使用EJB；开发有效的数据存取策略；设计简洁并且可维护性高的Web接口；设计高性能的J2EE应用程序等。本书的观点是完全独立的，面向问题而非规范，并以作者在生产实践中使用J2EE的实际经验为基础。阅读完本书之后，熟悉J2EE的基本概念但可能还没有任何J2EE使用经验的开发人员，将能够自信地尝试J2EE项目。经验丰富的设计师或开发人员将能够从本书以实用角度为出发点的J2EE体系结构与实现的讨论中受益，因而本书适用于Java设计师、具有J2EE经验的开发人员以及拥有J2EE基础知识并希望从事J2EE项目的Java开发人员。

版权贸易合同登记号：01-2003-0742

图书在版编目（CIP）数据

J2EE设计开发编程指南/（美）约翰逊（Johnson，R.）著；魏海萍等译. —北京：电子工业出版社，2003.7

ISBN 7-5053-8770-7

Ⅰ. J… Ⅱ. ①约… ②魏… Ⅲ. JAVA语言—程序设计 Ⅳ. TP312

中国版本图书馆CIP数据核字（2003）第041721号

责任编辑：春 丽 和 敬

印 刷：北京天竺颖华印刷厂

出版发行：电子工业出版社 http://www.phei.com.cn

北京市海淀区万寿路173信箱 邮编：100036

北京市海淀区翠微东里甲2号 邮编：100036

经 销：各地新华书店

开 本：787×1092 1/16 印张：39.375 字数：1000 千字

版 次：2003年7月第1版 2003年7月第1次印刷

定 价：64.00元

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换，若书店售缺，请与本社发行部联系。联系电话：（010）68279077

作 者 简 介

Rod Johnson是一名专攻可缩放Web应用的企业Java设计师。Rod花费两年时间为FT.com——欧洲最大的商业门户网站设计并实现了一个J2EE解决方案，后来经过长途跋涉到了Everest Base Camp，做了一个孩子的父亲，并撰写了本书。Rod希望感谢Tristan给他提供了难度最大的挑战。

Rod在悉尼大学通过主修音乐与计算机科学获得了文学士学位，在回到软件开发之后他获得了音乐学博士学位。由于有C和C++方面的基础，Rod自从Java与J2EE发布以来一直使用它们。他目前是JSR 154 Expert Group的成员，这是一个定义Servlet 2.4规范的专家组。

Rod已经给包括“Professional Java Server Programming（J2EE and J2EE 1.3 edition）”和“Professional Java Server Pages（2nd edition）”在内的其他Wrox出版物贡献过几章内容，并且是Wrox出版社的一名审稿专家。他出席过包括Times Java（2001于Mumbai）在内的国际会议，而且他的作品已被登载在java.sun.com站点上。

Rod Johnson是澳大利亚人，但他目前生活和工作在伦敦。

他的联系地址是expert@interface21.com。

简　介

笔者相信，J2EE是当今可用于企业软件开发的最佳平台。它结合了Java编程语言的各种优点和过去10多年中企业软件开发的种种教训。

然而，这一承诺未必完全得到实现。许多J2EE工程项目中的投资回报是令人失望的。传输系统的速度常常太慢，结构常常过于复杂。开发时间常常和业务需求的复杂性不成比例。

原因出在什么地方呢？与其说是由于J2EE中的缺陷所致，倒不如说是因为J2EE常常没有得到正确使用的缘故所致。而不正确地使用通常是由忽视现实问题的体系结构和开发的方法所引起的。起着主要作用的一个因素是许多J2EE出版物中过分强调各种J2EE规范，而忽略了人们使用这些规范来解决的各种现实问题。现实应用中常常出现的许多问题受到了忽视。

当阅读J2EE讨论论坛时，笔者强烈地感觉到，许多开发人员几乎没有找到自己的行动准则和方向，结果浪费了大量时间和精力。在许多情况中，这些开发人员具有多年的IT经历，但仍觉得掌握J2EE很困难。

问题不是缺少关于J2EE组件的信息。许多图书和万维网站点在描述小服务程序、企业Java组件（Enterprise JavaBean，简称EJB）等方面都做得非常出色，而且对JNDI、RMI和JMS之类的技术也讨论得很充分。

问题出在如何达到下一个水平——怎样获得这些构造材料并使用它们以便在一段合理的期限内构造出满足现实业务需求的应用。在这方面，笔者觉得现有的大部分J2EE作品起着一种阻碍作用，而不是帮助作用。J2EE图书的世界与企业软件项目的世界之间存在一道鸿沟，也就是说脱了节。

本书的目标就是解决这个脱节问题，并提供如何在实践中有效地使用J2EE的明确方向和行动准则。笔者将帮助读者解决J2EE的常见使用问题，以及避免在J2EE项目中常犯的高代价错误。笔者将向读者揭示各J2EE服务和API的复杂性，以便读者能够按时间和预算要求构造出尽可能简单的可能解决方案。笔者将采用一种注重实际经验和实际效果的方法，进而对J2EE正统方法在实践中未能实现正确结果的地方提出质疑，并推荐已得到实践证明的有效方法。

笔者觉得，没有任何一本现有图书实现了这一目标。最接近这一目标的图书或许是Prentice Hall所出版的“Core J2EE Patterns”一书（ISBN：0-13-064884-1），这本书的出版曾经引起不少人的激动，因为终于有了一本论述如何使用J2EE组件的图书。“Core J2EE Patterns”确实是一本好书，也是J2EE设计师和开发人员的一个宝贵资源。尤其是，它所使用的方法已经得到广泛接受，但它是一个Sun出版物，而且无法帮助反映“各方的策略”。

这本书也只关注各种J2EE标准，而极少关注使用真正服务器时所遇到的问题。它没有提供明确的行动准则：只是经常不偏不倚地给出各种非常不同的可替换“设计模式（Design Pattern）”，不做任何具体的分析。已经能够在这些“设计模式”之间自信地做出选择的读者从本书中几乎得不到什么收获。

笔者见过的现有出版物、示例应用和讨论论坛越多，就越坚信J2EE需要一剂注重实际效果的健康良药。J2EE是一个伟大的平台。遗憾的是，针对它而提倡的许多体系结构没有帮助解决许多常见的问题。许多J2EE示例应用（比如Sun公司的Java Pet Store）十分令人失望。它们没有面对现实问题。它们的性能非常差，而且它们的代码常常没有考虑现实问题，因而提供了没有太大价值的模型。

笔者还深深感觉到，J2EE的新手与已经使用J2EE构造企业系统的开发人员之间在见解方面存在着差距。笔者以前的一名同事使用了一个引人遐想的绝妙词汇“多瘤的”来形容已经掌握了一项技术的实际使用技巧同时又留下了许多疤痕的开发人员。虽然J2EE的新手看起来像是遵守清规戒律的J2EE传教士，但“多瘤的”开发人员有所不同。他们不得不抛弃一些意识形态上的包袱来实现必要的功能度或达到足够的性能。像笔者的同事和笔者本人一样，他们已经发现，现实粗暴地改变了最初的想像。

在本书中，笔者将利用自己的亲身经历和行业知识帮助读者设计并开发出切实可行的解决方案，同时又不需要读者经历一个痛苦的过程来发现J2EE理论与现实之间的差距。

J2EE的神秘性

笔者以为，导致人们对J2EE失望的原因通常可以追溯到几个常见的神秘之处，而这几个神秘之处又证实了开发项目中的许多明显和隐含的假设。

- J2EE在应用服务器和数据库之间具有可移植性。
- J2EE是所有企业软件开发问题的最佳答案。如果使用非J2EE技术（比如RDBMS存储过程）能够解决的问题也能利用J2EE技术来解决，那么使用“纯”J2EE方法总是最好的。
- J2EE服务器负责性能和可缩放性，从而让开发人员能够集中精力实现业务逻辑。开发人员可以在很大程度上忽略J2EE“模式”的性能含义，并依赖产品中的可接受性能。
- J2EE能够让开发人员忘了数据存取和多线程化之类的低级问题，因为这些问题将由应用服务器透明地处理。
- 所有J2EE应用都应该使用企业Java组件（EJB），因为EJB是开发企业级应用的最基本J2EE技术。
- J2EE方面的任何问题不久将由更先进的J2EE应用服务器来解决。

下面，让我们来看一看上述每一个神秘之处。

可移植性是J2EE平台的一大馈赠。正如我们将要看到的，可移植性可以在实际应用中得以实现，但这不是J2EE的本质所在。绝大部分工程项目的需求是构造这样一个应用：它很好地解决一个目标平台上的某一个特定问题。在一个平台上运行很差的应用将永远不会被移植到其他平台（该应用可以被移植到运行在硬件威力更强的计算机上的另一个操作系统来获得足够的性能，但这不是专业开发人员所期望的那种可移植性）。

J2EE正统观念认为，应用在J2EE应用服务器之间应该是可移植的，并且必须能够处理不同的数据库。这两大目标之间的区别是非常重要的，有时也被忽视。应用服务器之间的可

移植性可以实现商业价值，因此常常也是一个现实的目标。数据库之间的可移植性比较容易实现，因此常常没有太大的商业价值。

可移植性通常被误认为“代码可移植性”：获得应用程序并能够无修改地在另一个平台上运行它的能力。笔者以为，这是一个需要付出极高代价的误解。对总代码可移植性的幼稚强调往往会在生产效率损失和不令人满意的交付方面导致严重的后果。尽管“编写一次，到处运行（WORA）”是Java本身所关注的一个现实目标，但同样也是一个适合于企业软件开发的危险口号，视资源的范围而定。

笔者不是在谈论开发“缩小包装式（shrink-wrapped）”构件（通常指EJB）的少数工程项目。这个颇具吸引力的概念仍需要到市场中得到验证。而且，笔者还见过一个以这两者为目标的重要构件：应用服务器可移植性（在这种情况下有意义）和数据库可移植性（几乎肯定会比它所值得的更麻烦）。

笔者更喜欢“设计一次，在任何地方重新实现几个接口（DORAFIA）”的口号。笔者承认，这句口号不那么有吸引力，而且让Java开发人员遵守它也是不可能的。但是，这种比较注重实效的方法在窗口化系统之类的其他领域内得到了广泛应用。

可移植性的神秘已经导致人们广泛地认为J2EE不能使用当今关系数据库的能力，而只能使用它们作为转储存储器。这种观点在现实生活中造成了极坏的影响。

这并不是说，笔者不相信J2EE应用能够或者应该是可移植的。笔者只是在说明一种更注重实效、更实际的可移植性观点。我们可以把J2EE应用设计成能够被轻松地移植，我们不能用诸如.NET之类的专有技术做同样的事情。

想像到J2EE是企业体系结构发展的最后阶段是十分令人愉快的；对象技术和Java语言的应用终于解决了10多年来困扰着业界的问题。遗憾的是，这不是现实，尽管在J2EE开发的许多方法中暗示了这一点。J2EE建立在早于它之前出现的许多技术之上。它只是向前进了一步，但不是最后一步，而且没有解决企业软件开发的所有问题。

对可移植性的过分强调以及这种以J2EE为中心的态度，已经导致这样一种假设：如果在标准J2EE中不能做某一件事情，那么做这件事情将是一个设计错误。这种假设甚至正在蔓延到随着EJB QL的引进而出现的EJB规范中（EJB QL是一种可移植但还不太成熟的查询语言，并且与可用于绝大多数J2EE应用的熟悉、成熟而又标准的SQL相比显得更复杂，但威力却更小）。

笔者把J2EE服务器看做一组企业资源（比如数据库）的指挥者。一个好的指挥对任何表演来说都是不可或缺的。但是，指挥不应该试图演奏各个乐器，而是应该把这项工作留给技巧熟练的专业人员。

最危险的神秘性或许是这样一种观点：J2EE是能够产生好的性能和可缩放性的一条方便的途径，同时其效率是一个比已得到认可的J2EE“设计模式”更无需担心的问题。这种观点会导致幼稚和效率不高的设计。这是非常遗憾的，因为Java业界之外的人对Java一直有这样一种恐惧心理：Java的性能极差。现在，事实表明Java语言提供了很好的性能，但有些流行的J2EE“设计模式”仍提供非常差的性能。

我们无法假设应用服务器能够负责性能和可缩放性。事实上，J2EE给我们提供了捆绑

J2EE应用服务器和数据库所需要的全部绳索。由于最佳性能一直是软件开发的主要追求目标，所以我们一直在用C和汇编语言编写Web应用。但是，性能对现实应用的商业价值是至关重要的。我们不能依赖Moore规则来让我们利用更快的硬件去解决性能问题。无论硬件威力有多么强大，出现性能问题的可能性始终是存在的。

J2EE服务器应该透明地处理数据存取之类的低级细节这一想法是非常有吸引力的。有时，这是可达到的，但会十分危险。另外，让我们来看一看关系数据库的例子。处于领先地位的企业级关系数据库管理系统（RDBMS）Oracle使用了一种完全不同于其他任何RDBMS产品的方式来处理加锁。使用粗糙还是精细事务意味着其性能在不同数据库之间有很大的差别。这就是说，“可移植性”会是虚假的，因为相同的代码在不同的RDBMS中可能会表现出很大差异。

Oracle和其他领先产品的价格都十分高昂，而且拥有令人印象深刻的能力。我们经常希望（或者需要）直接利用这些能力。J2EE在诸如事务管理和连接共享之类的基础结构性服务方面提供了十分有价值的标准化，但在任何时候，我们都将不放弃各种厚厚的RDBMS产品说明手册。

“J2EE = EJB”神秘性会导致代价高昂的错误。EJB是一种复杂的技术，虽然很好地解决了一些问题，但在许多情况下也增添了比其商业价值更大的复杂性。笔者觉得，大多数图书都忽视了EJB的非常现实的缺点，而鼓励读者自动使用EJB。在本书中，笔者将冷静地分析EJB的长处和弱点，并提供关于何时使用EJB的明确准则。

允许所使用的技术（J2EE和其他任何一种技术）来确定一个业务问题的解决方法往往会导致很差的结果。此类错误的例子包括决定业务逻辑应该始终用EJB来实现，即决定实体组件是实现数据存取的单一方法。事实上，只有J2EE组件的一个很小子集（笔者认为还应该包括服务器小程序和无状态会话EJB），才是大多数J2EE应用的核心。其他J2EE组件的价值在很大程度上取决于待解决的问题。

笔者主张一种问题驱动而非技术驱动的方法（Sun公司的“J2EE Blueprints”通过建议一种J2EE技术驱动的方法，可能已实际造成了不良的影响）。尽管我们应该努力避免重复发明已有的东西，但是盲目地遵从我们绝不应该亲自实现服务器能够实现（尽管是无效率地实现）的东西这一正统观念将会付出极高的代价。用来处理事务管理等的核心J2EE基础结构是一个天赐之物，但这并不是说各种J2EE规范中所描述的所有服务都是天赐之物。

有些人将会争辩说所有这些问题不久将会得到解决，因为J2EE应用服务器变得越来越先进。例如，Container-Managed Persistence（容器管理式持久性，简称MCP）实体组件（Entity bean）的超高效实现将证明它比使用原始SQL的RDBMS存取具有更快的速度。这是幼稚的，并具有不可接受的风险。在IT行业中，几乎没有为信心留有什么地方。制定决策必须依据已得到证明的确凿证据，而且信心可能会用错地方。

现在，人们仍在激烈地争论着这样一个问题：J2EE的某些特性（比如实体组件）在许多情况中永远都无法像某些替代方法那样管用。而且，“理想中的乐土”仍然很遥远。例如，实体组件在1999年被首次引进到EJB规范中时，不久就提供了卓越的性能。然而，接下来的两年暴露了该原始实体组件模型中的严重缺陷。现在，EJB 2.0规范中的各种彻底变革仍有待证明，而且EJB 2.1规范已正在设法解决EJB 2.0规范中的遗留问题。

本书的不同之处

首先，本书的观点是独立的，以笔者和同事在生产实践中使用J2EE的经验为基础。笔者不打算布道。笔者主张使用J2EE，但警告读者谨防J2EE正统方法。

其次，本书的焦点是注重实际经验。笔者想帮助读者使用J2EE实现高性价比的应用。本书的目标是揭开J2EE开发的神秘性。它解释如何使用J2EE技术来减少而不是增加复杂性。尽管笔者并不把焦点集中在任何一个单独的应用服务器上，但将讨论读者使用实际产品时可能会遇到的一些问题。本书将不回避各种J2EE规范没有自然解决的现实问题。例如，我们在EJB层中怎样使用Singleton设计模式？在EJB层中应该怎样做日志记录？

本书不打算包括J2EE的全部情况，只打算解释解决常见问题的有效方法。例如，它将重点介绍如何与关系数据库一起使用J2EE，因为大多数J2EE开发人员都会面对O/R映射问题。一般说来，本书只打算在解决最常见的问题方面提供大量的帮助。

在全书中，我们将只着眼于一个统一的示例应用。在讨论每个问题时，我们不是使用一个不切实际的抽象示例，而是着眼于一个规模较大而又较注重实际的完整示例的一小部分。这个示例应用就是一个联机售票应用。它的设计目的不是为了举例说明特定的J2EE技术（像许多示例应用那样），而是为了说明J2EE设计师和开发人员所面对的常见问题。

本书注重的是质量、可维护性和生产效率。

这是笔者在处理自己的第一个J2EE工程项目时希望自己能够备有的图书。这样，它将会节省笔者大量的努力，也将会节省笔者雇主大量的金钱。

笔者采用的方法

本书是面向问题的，而不是面向规范的。和许多论述J2EE的其他图书不同，本书不打算全面介绍所有的服务和API，而是承认并非J2EE的所有部分都是同样有用的，或者说对所有开发人员都是有意义的，而且将讨论的重点放在构造典型解决方案中所用到的那些部分上。

软件设计与其说是科学不如说是一门艺术。J2EE的丰富性意味着寻找出同一个问题的多个有效解决方案（和许多坏的解决方案）常常是有可能的。尽管笔者尽了最大努力来解释自己的观点（或偏见），但本书自然也反映了笔者使用J2EE的经验和对待J2EE使用的看法。笔者介绍了自己觉得非常管用的一种方法。但是，这并不意味着它是惟一有效的方法。

本书大体上反映了笔者对待软件开发的如下看法：

- 笔者努力避免宗教式的立场。笔者永远理解不了有那么多的开发人员花费那么多的经历和热情致力于UseNet上的激烈争论。这实在没有什么好处可言。
- 笔者是一名实用主义者。笔者关心结果胜于意识形态。当处理一个工程项目时，笔者所追求的主要目标是按时和按预算交付一个高质量的结果。笔者所使用的技术是实现这个目标的一种工具，而不是结果本身。
- 笔者认为OO原理应该加强J2EE开发。
- 笔者认为可维护性对任何一个已交付项目的价值是至关重要的。

为了保持与这一实用主义方法相一致，笔者将经常引用Pareto Principle（巴累托定律），该定律说少量的原因（20%）担负结果的绝大部分（80%）。巴累托定律（最初起源于经济学）非常适用于实际的软件工程，而且我们在动手处理J2EE工程项目时将会不断地遇到它。例如，它可以提醒我们，设法解决某一特定领域中的所有问题会比只解决大多数实际应用中的各种要紧问题困难得多（而且性价比低得多）。

笔者的方法反映了Extreme Programming（极限程序设计，简称XP）的一些教训。笔者是一名方法学怀疑论者，不打算大肆宣传XP。这不是一本论述XP的图书，但笔者觉得XP给J2EE理论提供了一个宝贵的补充。尤其是，我们将会看到下列定律的价值：

- 简单性。XP专业人员主张做“可能管用的最简单事情”。
- 避免浪费精力。XP专业人员不添加他们可能永远不需要的功能。这种方法用首字母缩写词YAGNI（You Aren't Going to Need It）来表示。
- 重点放在整个开发过程中的测试上。

读者对象

本书的阅读对象是Java设计师、已经使用过J2EE的开发人员以及拥有J2EE基础知识而又希望从事J2EE项目的Java开发人员。

本书不是EJB、小服务程序、JSP和J2EE的入门读物。不熟悉这些领域的读者在阅读本书时可能需要参考一本综合性参考书（参见下文的推荐读物）。

本书目标

本书的目标是让读者能够轻松自如地制定J2EE开发的体系结构决策与实现决策。

在阅读完本书之后，熟悉J2EE的基本概念但可能还没有任何J2EE使用经验的开发人员，将能够自信地尝试J2EE项目。经验丰富的设计师或开发人员，将能够从本书以实用角度为出发点的J2EE体系结构与实现的讨论中受益。论述项目选择、测试和工具的各章节对正试图了解采用J2EE技术有什么影响的经理们特别有用。

本书内容

本书所涉及的内容包括：

- 如何做出关键性的J2EE体系结构选择，比如是否使用EJB以及在何处实现业务逻辑。
- J2EE Web技术，以及Model-View-Controller（模型－视图－控制器，简称MVC）体系结构模式在Web应用中的有效使用。
- 如何有效地使用EJB 2.0，其中包括引进含有本地接口的EJB所造成的各种影响。
- 如何选择应用服务器。
- 对J2EE开发至关重要的OO开发原理。
- 在J2EE应用中使用Java组件（JavaBean），以及这种使用如何帮助开发可维护与可移

植的应用。

- 在J2EE应用中使用XML和XSLT。
- J2EE事务管理。
- 如何在J2EE应用中有效地访问关系数据库。
- 如何使用通用的基础结构代码来解决常见问题，并保证应用代码只关注问题区内的业务逻辑。
- 如何测试J2EE应用，特别是如何测试Web应用。
- 如何设计具有满意性能的J2EE应用，以及如何改善现有应用的性能。
- 包装和部署J2EE应用。
- 主流应用服务器（比如BEA WebLogic和Oracle 9 Application Server）的特征，而这些特征可能会影响我们设计J2EE应用的方式。
- 了解可缩放性的基础性设计选择的各种影响。

本书还包括和讨论了能够在读者的应用中使用的大量通用基础代码。

阅读本书的前提知识

阅读本书的前提知识包括：

- 足够的Java语言技能；
- OO原理的牢固掌握；
- 对经典OO设计模式的熟悉；
- 对JSP的一定熟悉；
- 对Web和分布式对象协议（如HTTP和IIOP）的熟悉；
- 对J2EE的基础（如RMI、JDBC和JNDI）的了解。

再具有下列知识是最理想的：

- 关系数据库基础知识；
- 对事务概念（ACID属性和隔离层次）的了解；
- 对XML和XSLT的基本了解；
- 对UMI，尤其是对类和序列图表的基本了解；
- 对基本计算机科学概念（如数据结构）的熟悉。

推荐读物

本书参考了由Addison Wesley出版的经典图书“Design Patterns: Elements of Reusable Object-Oriented Software”一书（ISBN 0-201-63361-2）中所讨论的23个设计模式。虽然这本经典著作中所介绍的许多模式只不过编纂了任何一名经验丰富的OO专业人员的设计习惯，但为设计师提供了一种公用语言，并且是任何一名需要提高技能的开发人员的必读之物。本书假设读者已经阅读过并理解了这本经典著作。

本书还使用了由Prentice Hall出版的“Core J2EE Patterns：Best Practices and Design

Strategies”一书（ISBN 0-13-064884-1）中所介绍的设计模型名称，因为这些模式名称已经得到广泛的接受。虽然不必参考“Core J2EE Patterns”一书也能阅读本书，但建议读者读一读这本书。不过需要注意的是，笔者在几个方面推荐了一种不同的方法。另外，“Core J2EE Patterns”一书的某些章节，特别是和实体组件（Entity bean）有关的那些章节在EJB 2.0规范发布之后已经过时。

另外，本书的一些J2EE设计模式参考了由Wiley出版的“EJB Design Patterns”一书（ISBN 0-471-20831-0）。同样，建议读者读一读这本书，尽管它不是必读的背景读物。

阅读一本论述J2EE 1.3平台的好参考书也是很有必要的。笔者推荐由Wrox Press出版的“Professional Java Server Programming J2EE 1.3 Edition”一书（ISBN 1-861005-37-7，中文版《J2EE编程指南（1.3版）》已由电子工业出版社出版）。由Ed Roman编写和Wiley出版的“Mastering Enterprise JavaBeans（Second Edition）”（ISBN 0-471-41711-4）也是一本论述EJB的好书。

读者可以从http://java.sun.com/j2ee/sdk_1.3/techdocs/api/index.html站点上联机地获得针对J2EE 1.3平台的完整Javadoc。

最后，所有专业的J2EE设计师和开发人员都应该参考定义J2EE 1.3平台的各种规范，这些规范可以从http://java.sun.com/j2ee/download.html站点上获得。在本书中，笔者将参考下列规范的相关部分（比如EJB 17.4.1节）：

- J2EE 1.3
- EJB 2.0
- Servlet 2.3
- JSP 1.2

在相关的地方，笔者还将参考现在仍处于公开草案阶段并且也可以从Sun站点上获得的各种J2EE 1.4规范版本：EJB 2.1、Servlet 2.4和JSP 2.0。

使用本书的软硬件需求

为了运行本书中的示例，读者将需要：

- Java 2 Platform, Standard Edition SDK v1.3或以上版本。我们使用了Sun SDK 1.3.1_02来运行所有样本代码。
- 一个支持J2EE 1.3的应用服务器。我们为本书中的示例应用使用了JBoss 3.0.0。
- 一个RDBMS。我们为示例应用使用了Oracle 8.1.7i。读者可以从Oracle站点上获得“Personal Edition”的一个免费评估拷贝。要想使用一个不同于Oracle 8.1.7或以上版本的数据库，读者将需要对示例应用做一些修改（文中已明确标出）。

要想运行示例应用，读者将需要下列第三方库：

- Apache Log4j 1.2
- JSP Standard Tag Library（JSTL）1.0的一个实现

要想使用第3章中所讨论的那些测试策略，读者将需要如下产品：

- JUnit 3.7或以上版本

- Apache Cactus J2EE测试框架

要想运行第13章中的所有Web内容生成示例，读者将需要如下产品：

- Apache Velocity 1.3
- Lutris XMLC 2.1
- iText PDF生成库
- Domify XML生成库

关于如何安装和配置最后这4个产品的信息，请参见附录A。

上面所讨论的所有第三方产品和库都是免费的，并且是开放源代码的。

要想构造源代码，读者将需要Apache Ant 1.4.1或较新版本，其中包括可选的任务支持。

本书中所有示例的完整源代码可以从我们的Web站点http://www.wrox.com/上通过下载来得到。下载有两个版本：一个版本含有上面讨论的所有库，而另一个版本是一个小得多的捆绑，且只含有本书中所讨论的源代码和编译代码。如果读者已经拥有或者希望下载上面所列举的那些第三方产品，只需下载那个较小的版本即可。

本书中的约定

为了帮助读者从正文中获得最大的信息量和跟踪正在发生的事情，我们在整本书中使用了许多约定。

这含有不应该忘记的重要信息，而且这些信息与周围的正文直接相关。

这种背景样式用于当前讨论的旁注。

至于正文中的其他样式，本书使用了如下约定。

- 当介绍重要单词时，给出了它们的原文。
- 使用后面的样式表示键盘上的键击：Ctrl-K。
- 使用后面的样式表示正文中的文件名、代码以及用户界面上的文字和URL：persistence.properties。

本书使用两种方式表示代码：

```
In our code examples, the code foreground style shows new, important, pertinent
Code.
While code background shows code that's less important in the present context, or
Has been seen before.
```

客户支持

我们始终十分重视听取读者的反馈意见，而且希望知道读者对本书的看法：喜欢什么，不喜欢什么，希望我们下次在什么方面做得更好。读者可以把自己的意见发给我们：给feedback@wrox.com发送电子邮件。请务必在邮件中注明书名。

如何下载本书的示例代码

当访问Wrox站点http://www.wrox.com/时，通过我们的Search工具，或者通过使用书名列表之一，简单地找出本书的书名。然后，单击Code栏中的Download，或者单击本书的详细信息页上的Download Code。

可以从我们的站点中下载的文件已经使用WinZip进行过归档。当读者已经把那些附件保存到自己的硬盘驱动器上的一个文件夹中时，需要使用WinZip之类的解压缩程序抽取那些文件。当抽取那些文件时，代码通常被抽取到章文件夹中。当开始抽取过程时，读者需要确保自己的软件（如WinZip）被设置成使用文件夹名。

勘误表

我们已经尽了一切力量保证正文或代码中没有任何错误。但是，人无完人，出错是在所难免的。如果读者在我们的某本书中发现了错误之处，比如拼写错误或故障代码，我们将非常感激读者提供反馈信息。通过发送勘误表，你可以节省其他读者受挫的时间，当然也可帮助我们提供更高质量的信息。请直接把这些信息以电子邮件的形式发送给support@wrox.com，你的信息将接受检查，如果正确，这些信息将被张贴到用于本书的勘误表页上，或用在本书的后续版本中。

要想查找Web站点上的勘误表，请转到http://www.wrox.com，并通过我们的Advanced Search工具或书名列表，简单地找出书名。然后，单击Book Errata链接，该链接在本书的详细信息页上位于封面图形的下面。

p2p.wrox.com

要想与作者和同行讨论，请加入P2P邮件表。除了我们的一对一电子邮件支持系统之外，我们的特有系统还提供关于邮件表、论坛和新闻组的programmer to programmer™联系人。如果张贴一个查询到P2P上，读者可以确信你的问题正被许多Wrox作者和当时正在我们的邮件表上的其他业界专家所检查。不仅在阅读本书期间，而且在开发自己的应用期间，读者都将会在p2p.wrox.com上找到许多对自己有帮助的不同列表。

要想订阅一个邮件表，只需按如下步骤进行操作即可：

1. 转到http://p2p.wrox.com/。
2. 从左菜单栏上选择合适的类别。
3. 单击希望加入的邮件表。
4. 按照说明来订阅和填写你的电子邮件地址和密码。
5. 回复你所接收到的确认电子邮件。
6. 使用订阅管理器来加入更多的列表，并设置你的电子邮件参数选择。

本系统为什么提供最佳支持

读者可以选择加入那些邮件表，也可以选择将它们作为一个周文摘来接收。如果没有时间或便利工具来接收邮件表，读者可以搜索我们的存档文件。宣传和垃圾邮件已被删除，而且你自己的电子邮件地址将由特有的Lyris系统来保护。关于加入或离开邮件表的查询以及关于邮件表的其他任何一般性查询，都应该被发送给listsupport@wrox.com。

目 录