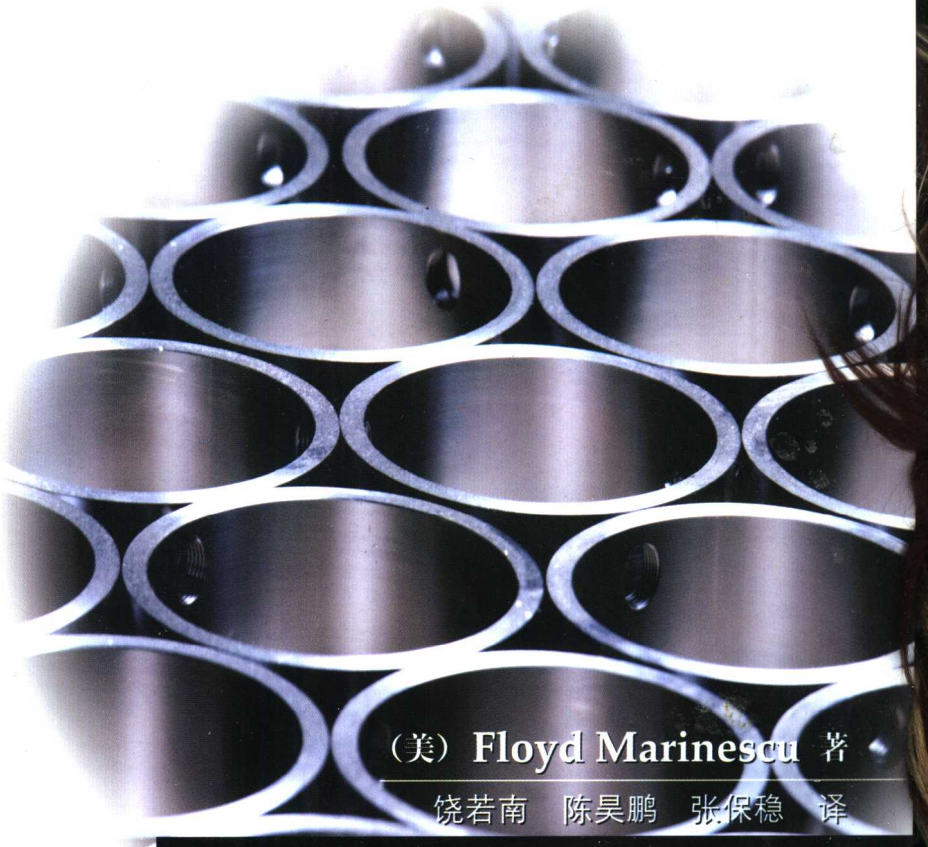
HZ BOOKS

Sun公司核心技术丛书

EJB 设计模式

EJB Design Patterns:
Advanced Patterns, Processes, and Idioms



(美) Floyd Marinescu 著

饶若南 陈昊鹏 张保稳 译

机械工业出版社 China Machine Press

中信出版社 CITIC PUBLISHING HOUSE

Sun公司核心技术丛书

EJB设计模式

（美） Floyd Marinescu 著

饶若南 陈昊鹏 张保稳 译

机械工业出版社
China Machine Press

中信出版社
CITIC PUBLISHING HOUSE

本书作者是世界上EJB设计模式领域内的卓越专家，他领导着EJB设计模式项目。本书不仅从理论上对设计模式在EJB上的应用进行了深入的探讨，而且通过实例展示了EJB设计模式的魅力。本书提出的模式都可以应用于实际项目中。阅读本书可以极大地拓展开发人员的思路，大幅度地提高开发人员构建应用系统的水平，提高代码的重用性，从而提高代码的质量。本书适合有一定经验的应用开发人员参考。

Floyd Marinescu: EJB Design Patterns: Advanced Patterns, Processes, and Idioms (ISBN: 0-471-20831-0).

Authorized translation from the English language edition published by John Wiley & Sons, Inc.

本书版权登记号：图字：01-2002-4239

图书在版编目（CIP）数据

EJB设计模式 /（美）马林纳斯卡（Marinescu, F.）著；饶若南等译. -北京：机械工业出版社，2004.1

（Sun公司核心技术丛书）

书名原文：EJB Design Patterns: Advanced Patterns, Processes, and Idioms

ISBN 7-111-13032-4

Ⅰ.E… Ⅱ.①马… ②饶… Ⅲ.Java语言－程序设计 Ⅳ.TP312

中国版本图书馆CIP数据核字（2003）第080763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）

责任编辑：吴 怡

北京牛山世兴印刷厂印刷·新华书店北京发行所发行

2004年1月第1版第1次印刷

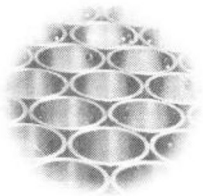
787mm×1092mm 1/16·14印张·1插页

印数：0 001-5 000册

定价：29.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线电话:（010）68326294



译者序

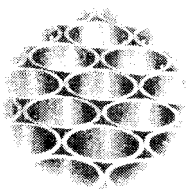
随着J2EE/EJB技术的日益流行，越来越多的企业选择使用J2EE/EJB技术来构建应用系统。但是，J2EE/EJB技术比较复杂，如何构建高效、安全和可靠的基于J2EE/EJB的企业应用系统是软件开发者要面临的一个挑战。很多有经验的EJB系统开发人员在他们的实践过程中总结出了许多可以提高系统性能的用于EJB编程的可复用方法，从而形成了EJB设计模式。这些设计模式可以帮助其他的EJB开发人员提高项目的质量，缩短项目的开发周期。但是EJB设计模式种类众多，所针对的问题各不相同，在解决问题时的侧重点也不尽相同。因此需要对它们进行分类整理，综合分析，使EJB开发人员能够在实际项目中选择最合适的设计模式。

本书正是以这种方法来介绍EJB设计模式的。这本书分为两大部分，第一部分介绍了EJB设计模式语言，包括系统架构设计模式、层间数据传输模式、事务和持久性模式、客户端EJB交互模式以及主键生成策略等；第二部分介绍了从需求到模式驱动的设计、用Ant编译与用JUnit进行单元测试的EJB开发过程、实体Bean的替代物以及众多的灵活小巧的设计技巧等。这本书通俗易懂，书中述及的EJB设计模式都是从实际的EJB项目中抽取出来的，所以它们非常实用。

本书的作者Floyd Marinescu是EJB设计模式方面的著名专家，他领导了EJB设计模式项目，并且和另外一名EJB领域的专家Ed Roman一起成立了一家培训和咨询公司，以帮助开发者掌握J2EE技术。

我们在翻译本书的过程中力求忠于原著，对于本书中出现的大量的专业术语尽量遵循标准的译法，并在有可能引起歧义之处标注了英文原文，以方便读者对照理解。

全书的翻译由饶若楠、陈昊鹏和张保稳合作完成。由于我们水平有限，书中出现错误与不妥之处在所难免，恳请读者批评指正。



序　言

许多具有良好架构的EJB项目都使用了设计模式，也许有的开发者当时并没有意识到在使用设计模式。开发者在项目开发过程中经常会构思出一些绝妙方案，但他们并没有意识到这些最优方案实际上就是设计模式（可复用的程序设计方法），而这些设计模式对其他开发者的项目也颇有益处。

这就是本书所述及的EJB设计模式的妙处所在——它们是从真实的EJB项目中抽取的实用的模式。J2EE开发人员可以到TheServerSide.com上与他人分享各种设计模式。我们将作为一个社团在一起工作，不断充实这些模式，以飨读者。

Floyd Marinescu是世界著名的EJB设计模式专家，是他领导着EJB设计模式项目。Floyd和我已经共事多年。我们成立了The MiddleWare Company（中间件公司），旨在进行培训和咨询业务，帮助开发者掌握企业Java技术。

我们公司为实际的项目提供咨询，帮助提高设计模式的质量。我们还开设针对开发人员的EJB设计模式培训课程，而且从这些课程中获得的反馈信息显著提高了本书的质量。

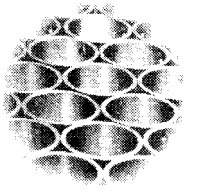
在本书中，Floyd将向大家展示若干EJB设计模式，它们可以用于提高EJB项目的质量。恰当应用这些设计模式可以提高体系结构的设计水平，提高代码的复用性和质量，而且使所架构的系统对于那些熟悉这些模式的开发人员来讲更容易理解。

本书最大的优点是它易于理解。Floyd选择了一种容易理解的叫做Alexandiran形式的风格来描写模式。这使得任何懂得用EJB编写程序的人都可以轻松地阅读本书（如果你还不了解EJB，可以读一下拙作《*Mastering Enterprise JavaBeans, Second Edition*》（精通EJB（第二版）），该书在书店和TheServerSide.com均可找到）。另外，你也可以参加EJB的培训课程，比如中间件公司提供的培训课程。

阅读本书你将获益匪浅。Floyd用了整整一年时间来处理复杂的EJB设计模式概念，这将对整个EJB社团都有益处。能和Floyd在这个项目上一起工作，我感到很荣幸，同时我也学到很多东西，相信读者也会感同身受。

Ed Roman

中间件公司CEO



前　言

这本书关系到软件开发人员的生活质量。无论开发者、系统架构师还是项目经理，在一天结束的时候，都会为在构建和发布设计精良的应用时没有产生代价高昂的错误、没有工作到深夜和没有经历那种长达数月的压力而感到高兴。我们都是普通人，都希望在一天结束的时候看到项目在按进度进行，并且回家后有充足的时间去干我们喜欢干的事情。

然而，当使用诸如Java 2 Enterprise Edition（J2EE）这类正处在发展阶段的新技术时，设计精良的应用并不容易做到。在这类相对较新的领域中，总是非常缺乏有关设计优良系统的知识。开发者要么效率低下地从头开始努力学习，要么每天都在项目上产生很多代价高昂的错误。如果没有一些有条理的最佳经验可遵循，EJB开发者的工作将十分艰苦。学习优良的设计方法在技术上对于初学者来说尤其困难，他们中的许多人在这之前从来就没有构建过分布式系统、也不清楚影响分布式系统设计的最基本需求。

更糟的是，EJB规范从一个版本到另一个版本的变化总是会给在设计优良的EJB系统时所使用的设计方式带来显著的变化。特别是随着EJB 2.0规范的引入，甚至在大多数最近出版的有关EJB的书中所讨论的多年来积累的最佳经验都将不再适用，或者达不到相同的目的了，使用这样的最佳经验可能导致设计出的系统非常拙劣。

我们编写本书（中间件公司的“EJB for Architects”课程也讲授这本书中所描述的模式）为的是传播精妙的设计思想，提高开发者的设计质量，从而提高开发者自己的生活质量，我们的最终目的是希望帮助读者学习当今业界中顶尖的设计方法，以便快速地设计出高效的、可扩展的和可维护的系统。

我们希望使用模式（Pattern）这种机制来给读者传授设计知识。

什么是模式

模式是对常见的重复出现问题的最佳解决方案。也就是说，模式说明并解释了在设计和实现应用时所产生的重要的或者是具有挑战性的问题，并且

讨论了对这个问题的最佳解决方案。随着时间的推移，模式开始体现出产生它的业界的集体智慧和经验。例如，本书中的模式呈现了成千上万的来自TheServerSide和业界的EJB开发者的智慧，他们的意见和评论为本书提供了很好的素材。

模式的优点

模式当然有很多的用处，但是下面这些优点对于帮助诸如J2EE这样的新软件平台尽快成熟具有重要意义：

为讨论设计问题提供高级语言。EJB开发者可以使用本书中的模式名字来高效地讨论实现细节。想象一下，描述一个应用是使用无状态会话外观（stateless session façade）模式构建的比设法解释会话bean怎样包装实体bean的所有语义要快许多。

预先提供了许多设计工作。一个描述完备的模式详细讨论了需要解决的问题，并且通过正反两方面的讨论，说明了问题应该如何解决，以及其他应该了解的问题。通过研读模式，可以预先讨论和考虑许多具有挑战性的并且是潜在的深层问题。

模式的组合有助于导出可复用的体系结构。模式总是在彼此之间互相引用和依赖。这种模式之间的连接创造了模式语言（pattern language）：即一系列互联的模式，它们整体上常常代表了应用的完整的体系结构。这样，当阅读本书时，某些互联模式的集合将形成可以被反复应用于多个项目的可复用的体系结构。

在本书中，我们重点选取了非常底层的与EJB相关的模式，也就是说，本书不关心那些可以应用于多种技术的通用的抽象模式，我们重点描述EJB模式，讨论与具体EJB相关的问题和复杂的实现方法。这样，在本书中，我们与许多其他的通常是说明一个模式的具体实现的模式书籍不同（那些模式不是与具体的项目相关的），我们的目标是为EJB开发者或架构师提供在基于EJB/J2EE的应用中使用这些模式所必需的所有信息。

模式的起源

对许多人来说，首次接触模式都是通过里程碑式的经典著作《设计模式：可复用面向对象软件的基础》（*Design Pattern: Elements of Reusable Object-Oriented Software*，Gamma,et al,1994，这本书已由机械工业出版社引进出版）。尽管这本书并不是第一本关于软件模式的书，但该书对于将模式的概

念和使用引入到了软件开发的主流中去起到了不可估量的作用。

模式的实际起源要远早于1994年。模式是Christopher Alexander第一个描述的，并且在20世纪70年代应用于城镇和建筑物的结构/体系中。在《*A Pattern Language*》（建筑模式语言，1977年）这本书中，Alexander写道："每一个模式都描述了在我们周围的环境中一次又一次地重复发生的问题，并且描述了对这个问题的解决方案的核心内容，这样我们就可以成百万次地使用这个解决方案而又不会老调重弹。"

模式是用来组织在生活中任何领域内的知识和解决方案的优秀方法，而不仅仅限于民用工程和软件设计领域。模式的结构和本性使它们能够很好地适用于对知识进行分类。好的模式不仅说明了解决问题的方法，而且以一种有助于解释问题的特征及其解决方案的形式组织起来。

本书中使用的模式描述风格

本书采用了一种与Alexander所使用的、称为Alexandrian 形式（Alexandrian form）的类似原始样式的风格来描述模式。我使用的格式为叙述风格，分为问题和解决方案等如下的几个段落：

- 语境（context）：描述有关模式应用语境。
- 问题（problem）：**一个问题，用来说明这个设计模式将要解决的难题。**
- 限制（forces）：详细解释语境和问题的若干段落，解释了在需要一个解决方案的工作中所存在的许多限制，这里读者将完全理解为什么需要这个模式。
- 解决方案（solution）：**用于介绍作为上面所描述问题的解决方案的模式。**
- 解决方案描述（solution description）：描述这个解决方案的若干段落，包括从正反两方面、从高层到低层对模式和在EJB中的实现问题的解释。

相关的模式

"相关的模式"部分会交叉引用这本书中的其他模式，或其他资源中相同或相似的模式，"参考文献"部分详细介绍了其他资源。

本书使用叙述风格而非要点陈述风格来描述模式，这使得在本书中用到的Alexandrian风格与大多数其他流行的软件模式书籍不同。本书的目标之一就是使有经验的架构师和初学者都可以读懂一个模式并且完全理解它。我

努力实现这个目标的方法是采用流畅的叙述风格，把读者由问题引导到解决方案，在所有的必要点上都一步一步地解释，强调的是以一种简单而愉快的方式学习模式。

在使用非Alexandrian风格时，往往会把模式分解为彼此互相分离的部分，其中每一部分都使用要点陈述风格进行描述。我认为采用这种风格是为了把尽可能多的信息打包在每一个要点中，这使得阅读和理解变得很困难，因为这些信息更适合于参考而非适合于最终阅读。因此这样的模式书籍更适合于参考，但是本书的目标是同时将模式教授给有经验的架构师以及没有经验的开发者，所以必须选择一种有助于学习的描述风格，Alexandrian风格是最佳之选。

本书的组织

本书在组织上可以分为两部分：第一部分“EJB模式语言”是EJB模式的目录，详细介绍了20种模式；第二部分“EJB设计和实现的最佳实践”包括围绕其他主题（例如模式的应用和实现基于EJB的实际系统等）讲授最佳方法的若干章节。我们的目标不仅是要给读者一个模式的目录，还要给读者掌握这些模式的知识和完成工作所必需的工具。在第二部分同时还包括一个用于描述实体bean的替代品的章节，它给EJB开发者提供了使用Java数据对象（Java Data Object, JDO）来持久化一个对象模型的前景，还有一章描述了粒度更细的，因为其太小而难以成为一个完整模式的设计技巧和策略。

本书的读者

本书深入讨论高级EJB话题，所以我们假设读者在读本书之前已经对EJB的基础知识有了很好的了解。我特别推荐Ed Roman的《*Mastering Enterprise JavaBeans, Second Edition*》（Wiley 2001），它是一本学习EJB基础知识和高级应用的优秀书籍。本书最初是Ed Roman的书中的一章，但是由于其内容不断地增加，于是我们决定将它独立成书。获得掌握本书内容所需背景的另一个途径是参加由中间件公司提供的Mastering EJB课程或者EJB for Architects课程来学习这本书中所有的模式。

尽管这本书要求读者要了解有关EJB的知识，但它并不是仅针对有经验的架构师的。这本书的风格也适合于入门级的读者。特别是，在这本书的撰

写过程中我们遵从了下面三个原则：

1) 某些刚刚读过Ed Roman的《*Mastering Enterprise JavaBeans, Second Edition*》或其他EJB书籍的读者应该能够在没有什么太多困难的情况下就可通读和掌握这些概念。

2) 本书内容的核心程度和技术深度即使对专家也颇有吸引力。

3) 本书的每一部分所回答的问题都比它提出的问题要多得多。

这三条原则是这本书的灵魂，原则3)可以保证我们提供在解释一个令人费解的话题时所必需的背景知识，而原则1)还使得本书重复了一些对EJB的介绍性的内容，原则2)则保证我们重点讨论那些在别的书中通常没有提到的、然而却是最有用和最重要的话题。

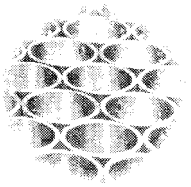
相关Web站点

与本书相伴的Web站点：www.theserverside.com/pattern/ejbpatterns，包含了这本书中运行和编译的源代码例子，以及一个读者论坛。这个Web站点将很有希望不断地发展下去，因为随着时间的推移，读者会不断地向这个Web站点加入更多的模式。

小结

本书与任何其他的模式书不同。你将不仅学习到很多有价值的、基础的模式，它们可以帮助你提高所编写的基于EJB的应用的质量，而且还将学习到怎样去接受这些模式的知识并以用例驱动的方式应用它们。同时，你将学习到许多关于如何去实现已经完成设计的应用的最佳方法。

如果本书对你的事业和你正在从事的项目的质量有所贡献的话，我将非常欣慰。如果本书能够通过提高你的项目的效率从而使你的整个生活质量有所提高、那么它就超额完成了任务。



致　谢

我要感谢ED Roman和Agnieszka Zdaniuk，没有他们的信任，本书不可能完成。

我要感谢Florin和Carmen Marinescu，他们很早就教我什么是重要的。

特别感谢Aravind Krishnaswamy和Mark TurnbUll，他们帮助我履行了生活中的其他一些责任，使我可以专注地投入本书的写作。

文本贡献者

我要感谢Randy Stafford，他在第7章关于开发过程方面的贡献，感谢Craig Russell，他为非实体bean的章节提供了JDO方面的材料。

代码和模式思想的贡献者

Ricahrd Monson-Haefel提出了使用行集（RowSet）来进行数据传输。

Doug Bateman提出了为自动生成键使用存储过程的初始建议。

Steve Woodcock提出了EJB模式的UUID的概念和代码。

Stuart Charlton提出了通用属性访问的思想。

模式指南

我要感谢Markus Voelter和Ralph Johnson，并特别感谢Bobby Woolf，没有他们早期提出的关于模式形式的建议，本书的内容将十分混乱。

审阅者

没有TheServerSide.com J2EE社区成员的评论、建议、修正、质疑，本书不可能完成。他们用八个月的时间审阅了拙作。他们有：

Alex Tyurin, Alex Wologodzew, Allan Schweitz, Alun R. Butler, Andre Cesta, Andre Winssen, Andreas Krüger, Andy Stevens, Andy Turner, Ankur Kumar, Anthony Catalfano, Anuj Vohra, Anup Kumar Maliyackel, Aparna Walawalkar, Ashim Chakraborty, Babur Begg, Ben Beazley, Bill Ennis, Billy

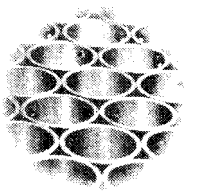
Newport, Blasius Lofi Dewanto, Bob Lee, Boris Melamed, Botnen Trygve, Brian Benton, Brian Dobby, Brian Walsh, Brian Weith, Carmine Scotto d'Antuono, Cecile Saint-Martin, Chad Vawter, Chandra Mora, Charles N. May, Colin He, Constantin Gonciulea, Cristina Belderrain, Curt Smith, Dan Bereczki, Dan Zainea, Daniel F. Burke, Daniel Massey, Darrow Kirkpatrick, Dave Churchville, David E. Jones, David Ezzio, David Ziegler, Dimitri Rakitine, Dimitrios Varsos, Dion Almaer, Doal Miller, Don Schaefer, Donnie Hale, Eduard Skhisov, Emmanuel Valentin, Engström Anders, Erez Nahir, Faisal , aveed, Fernando Bellas Permuy, FM Spruzen Simon, Forslöf Mats, Frank Robbins, Frank Sampson, Frank Stefan, Fried Hoeben, Gabriela Chiribau, Ganesh Ramani, Geert Mergan, Gene McKenna, Geoff Soutter, Gjino Bledar, Gunnar Eikman, Hai Hoang, Heng Ngee Mok, Hildebrando Arguello, Hossein S. Attar, Howard Katz, Huu-An Nguyen, Iain McCorquodale, J.D. Bertron, James Hicks, James Kelly, Janne Nykanen, Jean Safar, Jean-Pierre Belanger, Jeff Anderson, Jérôme Beau, Jesper Andersen, John Ipe, Jonathan Asbell, Jörg Winter, Joseph Sheinis, Juan-Francisco Borras-Correa, Julian Chee, Junaid Bhatra, Justin Leavesley, Justin Walsh, Ken Hoying, Ken Sizer, Krishnan Subramanian, Kristin Love, Kyle Brown, Lance Hankins, Larry Yang, Laura Fang, Laurent Ricu, Leo Shuster, M Heling, Madhu Gopinathan, Mark Buchner, Mark L. Stevens, Martin Squicciarini, Matt Mikulics, Mattias Fagerström, Mohan Radhakrishnan, Mohit Sehgal, Muhammad Farhat Kaleem, Muller Laszlo, Murray Knox, Nakamura Tadashi, Nicholas Jackson, Nick Minutello, Nick Smith, Niklas Eriksson, Oliver Kamps, Olivier Brand, Partha Nageswaran, Patrick Caulfield, Paul Wadmore, Paulo Ferreira de Moura Jr., Paulo Merson, Peter Miller, Pontus Hellgren, Raffaele Spazzoli, Rais Ahmed, Rajesh Jayaprakash, Reg Whitton, Richard Dedeyan, Rick Vogel, Robert McIntosh, Robert Nicholson, Robert O'Leary, Roger Rhoades, Roman Stepanenko, Samuel Santiago, Sashi Guduri, Scot McPhee, Scott Chen, Scott Stirling, Scott W. Ambler, Sébastien Couturiaux, Sergey Oreshko, Shorn Tolley, Simon Brown, Simon Harris, Simone Milani, Stefan Piesche, Stefan Tilkov, Stephan J. Schmidt, Steve Divers, Steve Hill, Steven Sagaert, Sun-Lai Chang, Tarek Hammoud, Taylor Cowan, Terry Griffey, Thanh C. Bryan, Therese Hermansson, Thierry Janaudy, Thomas Bohn, Toby Reyelts, Tom Wood, Tracy Milburn, Trond Andersen, Tyler Jewell, Udaya Kumar, Vaheesan

Selvarajah, Vincent Harcq, Yagiz Erkan, Yi Lin, and Yousef Syed。

最后，我要感谢那些对我的生活产生积极影响从而间接地为本书的出版作出贡献的人们，他们是：

Nitin Bharti, Sudeep Dutt, Morrissey, Calvin Broadus, George Kecskemeti, Johnny Marr, Geoff McGuire, Andre Young, Katerina Ilievska, Chogyam Trungpa, Siddhartha Gautama, Nadia Staltieri, Dale Carnegie, Lao-Tzu, David Gahan, Bogdan and Andre Cristescu, Umar Sheikh, Robert Smith, Ursula and Suzanna Lipstajn, James McDonald, Jacob Murphy, Olivia Horvath, Peter Coad, Mohandas K. Gandhi, Adib Saikali, Giacomo Casanova, Sasa Nikolic, Deanna Ciampa, Aravind Krishnaswamy, Mikola Michon, Mark Turnull, Laura Ilisie, Gregory Peres, Stuart Charlton, and Carlos Martinez.

关于作者



Floyd Marinescu是EJB设计模式方面的顶尖专家之一。他构建了TheServerSide.com——世界级领先的J2EE社团Web站点，并使其投入了运行。Floyd编写了大量的EJB设计模式，并且和社团一道努力工作，从而获得了EJB的最佳经验。他撰写了大量的论文，并在主要的Java会议频频亮相。Floyd同时也为中间件公司（www.middleware-company.com）工作，该公司是一家在EJB 和J2EE方面的培训和咨询公司。读者可以通过floyd@middleware-company. com和他取得联系。

关于为本书作出贡献的人

Randy Stafford是一个在软件开发的许多方面都很精通的专家，他在金融服务、数据库管理系统软件、医院、无线通信、交通、CASE和航天以及国防系统等多个信息技术的领域有15年的经验。他有作为许多公司的咨询师或者永久雇员的广泛的阅历，其中大公司有Travelers Express、Oracle、AMS和Martin Marietta，小公司有Gemstone、SynXis和Ascent Logic 企业。作为Smalltalk的开发者，他自1998年起就专注于面向对象和分布式系统的开发，并且自1995年起还加入了Web开发和电子商务的项目。

现在，作为IQNavigator公司的首席构架师，Stafford先生自1997年以来使用不同的J2EE应用服务器和Java ORB开发了8个分布式产品的Java应用。他是GemStone系统公司的J2EE范例FoodSmart的创始人和构造者。他也是GemStone关于设计J2EE应用的模式语言的设计者，是J2EE应用框架之一GemStone专业应用服务基础类的设计者。他曾在2000年夏回溯他的后5个J2EE应用的最初发行版，用Ant构建了它们的自动版本。从1998年初最初发布JUnit开始，他一直在使用它，并且从1995年开始，他还使用了JUnit的前身SmalltalkUnit。

Stafford先生毕业于Colorado州立大学，获得过应用数学的理学学士学位和计算机系的授课硕士学位。他在面向对象的仿真和系统工程文献中赫赫有名。

Craig Russell是Sun微系统公司的产品架构师，负责一个对象关系映像引擎——透明持久性的体系结构。在过去的30年中，他从事过企业规模的分布式事务和数据库系统的构造、设计和支持。

Craig 是JDO（Java Data Object）规范的领导者，该规范是针对以Java为中心的持久性的、由JCP（Java Community Process）作为一种Java规范提案进行管理。

目　录