J2EE Design Patterns Applied

实用J2EE设计模式编程指南

Craig A. Berry
〔美〕 John Carnell 等著
Matjaz B. Juric

邱仲潘 等译

乐思 WROX PROGRAMMER TO PROGRAMMER™

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

J2EE Design Patterns Applied

实用J2EE设计模式编程指南

Craig A. Berry
〔美〕 John Carnell 等著
Matjaz B. Juric

邱仲潘 等译

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内 容 提 要

模式已成为收集、规范和分析某些情景中常见问题的有效方法。本书介绍J2EE设计模式，及如何应用这些模式建立高质量应用程序，包括设计企业方案应用程序时使用的各种设计模式，并分章节介绍各个模式。本书有针对性地列举大量实用代码，以便读者理解和掌握J2EE企业开发中的常见问题及其解决方案。本书适用于J2EE开发人员。

版权贸易合同登记号：01-2002-4221

图书在版编目（CIP）数据

实用J2EE设计模式编程指南/（美）贝里（Berry，C. A.）等著；邱仲潘等译. —北京：电子工业出版社，2003.1

书名原文：J2EE Design Patterns Applied

ISBN 7-5053-8148-2

Ⅰ. 实… Ⅱ. ①贝… ②邱… Ⅲ. Java语言—程序设计 Ⅳ. TP312

中国版本图书馆CIP数据核字（2002）第091762号

责任编辑：马振萍

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：18.5　字数：470千字

版　　次：2003年1月第1版　2003年1月第1次印刷

定　　价：31.00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

Craig A. Berry

Craig A. Berry是Wrox出版公司的特约编辑、技术编辑、技术建筑师、作家，最近4年来参与了25本编程图书的工作，包括《Professional Java Server Programming J2EE》、《Professional EJB》、《Professional JMS Programming》、《Professional Java Servlets 2.3》和《Professional J2EE EAI》。Craig在Java和图书出版方面已经有丰富的经验，虽然不是最新Wrox Java出版物的策划者，但是可以在电影领域中得知他。他的E-mail地址为：craigb@wrox.com。

Craig编写了本书第4章。

John Carnell

John Carnell从12岁使用Commodore 64并投身计算机世界，是系统体系结构与设计方面的活跃作家和专业演讲者。John目前是Centare集团的系统建筑师，Centare是电子软件方案的一流供应商。John和妻子Janet一起住在威斯康星州Waukesha市，有一个儿子Christopher和两个宠物Lady Bug与Ginger。

他的E-mail地址为：john_carnell@yahoo.com。

献给我的妻子Janet：你是我们家庭的支柱，使我的生活中每一天都充满欢乐。

献给我的儿子Christopher：你是我的希望与梦想所在，看着你我可以在瞬间忘记自己的缺点与疑惑，体会到生活的美妙与特殊。我爱你们。

John编写了本书第3章。

Matjaz B. Juric

Matjaz B. Juric是计算机与信息科学博士和Maribor大学助理教授，参与了几个大型对象技术项目。他与IBM Java技术中心合作，进行了Java 2平台集成部分RMI-IIOP开发中的性能分析与优化工作。Matjaz合著了Wrox出版公司出版的《Professional J2EE EAI》和《Professional EJB》等书，并在剑桥大学出版社出版的《More Java Gems》一书中编写了其中一章。他在《Java Developer's Journal》、《Java Report》、《Java World》、《Web Services Journal》、《EAI Journal》和ACM杂志与OOPSLA、SIGS Java Development、Wrox Conferences、XML Europe、SCI等会议上均发表了文章。他还是个评论员、项目委员会委员和会议组织者。

谨将本书献给我的家人。特别感谢Wrox出版公司、Maribor大学的所有朋友和每个支持我的人。

Matjaz编写了本书第1章和第6章。

Meeraj Moidoo Kunnumpurath

Meeraj是EDS公司的高级信息专家，支持Chelsea FC，每周跑25英里，晚上吃金枪鱼和花椰菜。

Meeraj编写了本书第5章。

Nadia Nashi

Nadia最初是学建筑的，1993年在格林威治大学获得了建筑学的硕士文凭。然后她从设计实物对象转入设计虚拟对象，1994年在Westminster大学软件工程系取得硕士学位。Nadia是全日制的独立面向对象分析/设计师、Java顾问和软件开发人员。

谨献给我的丈夫Adrian和女儿Mariam。

Nadia编写了本书第2章和第7章。

Sasha Romanosky

Sasha取得加拿大Calgary大学电子与计算机工程学士学位，在计算机与Internet技术方面已经有七年多的经验，研究范围包括加密法、PKI、安全模式和基于角色的访问控制，热衷于Internet安全工作。

Sasha编写了本书第5章。

简　介

欢迎使用《实用J2EE设计模式编程指南》一书。顾名思义，本书介绍J2EE设计模式及其如何应用建立高质量应用程序。几年来，模式已经成为收集、规范和分析某些情境中常见问题的有效方法。J2EE平台设计模式在现有软件设计模式中增加了新元素。但是，这些设计模式不是独立存在的，需要在更大更复杂的框架中理解。

选择设计模式和将其变成实际方案并非易事。此外，采用一般设计模式和特殊的J2EE设计模式解决业务与技术要求有很大的难度。本书介绍如何用J2EE模式建立安全和可伸缩的J2EE应用程序，包括面向对象设计原则和实际实践。

本书不仅介绍J2EE企业开发中的常见问题及其解决方案，而且介绍如何运用到实际项目中。

本书适用对象

本书适合熟悉Java 2平台和企业版的开发人员更好地利用J2EE技术，包括：

- 让熟练的J2EE开发人员提高技术水平，将J2EE知识推向极致。
- 让J2EE建筑师设置系统规范、确定Java 2企业方案和将其集成到数据库、遗留系统等后台系统。

本书内容

本书包括设计企业方案应用程序时使用的各种设计模式，分章节介绍各个模式，并应用到设计与开发的不同领域。我们用案例介绍每个设计模式及其运用。本书的章节如下：

- **第1章：J2EE设计模式**
  本章概述模式，并特别介绍设计模式与J2EE。同时介绍一些准则，如怎样有效利用设计模式。
- **第2章：Web层设计模式**
  本章介绍设计与管理Web应用程序顶层的问题，介绍客户层的技术需求和如何用J2EE表示设计模式解决。
- **第3章：持久性框架设计模式**
  本章专门介绍用常见数据持久性设计模式与J2EE技术结合起来开发数据层。本章特别介绍持久框架及建立持久框架时要考虑的核心模式。
- **第4章：改进性能与伸缩性的设计模式**
  本章介绍几个改进性能与伸缩性的设计模式。
- **第5章：管理安全性的设计模式**
  本章介绍安全性模式及其好处。安全模式可以解决安全问题、J2EE并提供了解决已知编程问题的可靠技术的其他面向对象模式。

- **第6章：企业集成设计模式**

  本章介绍集成设计模式及其在J2EE标准中的运用。

- **第7章：复用性、可维护性与扩展性设计模式**

  本章介绍实现复用性、可维护性与扩展性的设计模式，然后介绍在一般性框架中演示这些方面的模式，包括J2EE框架情境中的一系列组件。

使用本书的要求

本书所有代码都是在Java 2平台，企业版SDK 1.3 Reference Implementation中测试的，可以从http://java.sun.com/j2ee/download.html下载。本书有几章涉及数据库访问，这些章节用到了开放源代码数据库服务器MySQL，可以从http://www.mysql.com/downloads/index.html免费下载。

规则

为了更好地阅读本书，我们在书中使用了几个规则。

例如：

黑体字表示重要的信息，与周围文本直接相关。

而背景样式表示当前内容的旁白。

文本样式如下：

- 引入单词时用高亮显示。
- 键盘单击显示为如下：Ctrl-K。
- 文本中的文件名和代码显示如下：persistance.properties。
- 用户界面与URL中的文本如下：Menu。

代码用两种方式显示：

```
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

我们非常希望收到读者的反馈，想了解您对本书的看法：喜欢哪里，不喜欢哪里，哪些地方可以改进。可以发信到feedback@wrox.com。您的反馈消息中一定要注明书名。

如何下载书中样本代码

登录Wrox站点http://www.wrox.com/时，只要通过Search功能找到本书或使用书名清单。单击Code列的Download或单击书中细节页的Download Code。

可以从站点下载的文件已经存成WinZip。将附件保存到硬盘文件夹中后，需要用WinZip或PKUnzip之类的解压缩程序解压缩文件。解压缩文件时，代码通常解压缩到各章文件夹中。

开始解压缩时，一定要将软件（WinZip、PKUnzip）等等设置成用户文件夹名称。

勘误

虽然我们尽力保证文件和代码中不出现错误，但错漏之处仍在所难免。如果发现书中的拼写错误或代码错误，请一定反馈，帮助消除读者的困惑并帮助我们提供更高质量的图书。可以发E-mail到support@wrox.com，我们会检查读者的信息，如果正确则会发表到该书的勘误页，或在该书的修订本中采用。

要寻找Web站点的勘误页，登录http://www.wrox.com/并通过Advanced Search或书名清单找到该书。单击书中细节页的封面图形下面的Book Errata链接。

E-mail支持

如果想向了解图书细节的专家直接了解书中问题，可以发E-mail到support@wrox.com，在E-mail主题字段中提供书名和ISBN号的最后四位（本书为5288）与页号，典型E-mail应包含下列内容：

- 在主题字段中提供书名和ISBN号的最后四位与有问题的页号。
- 在消息体中包含你的姓名、联系信息和问题。

我们不会向你发垃圾邮件。但我们需要细节信息，节省双方的时间。发送E-mail消息时，它会经过下列支持链：

- 客户支持——消息发送给客户支持人员，他们是第一读者。他们具有常见问题的文档，会立即回答基于该书或Web站点的立即问题。
- 编辑——较深的问题会转交给负责该书的技术编辑。他们具有编程语言或特定产品的经验，可以回答该主题的详细技术问题。
- 作者——最后，如果编辑无法回答你的问题，则会把它转交给作者。我们尽量不分散作者的创作精力，但我们乐意把特定问题转交给他们。所有Wrox作者都会帮助支持自己的书。他们会向客户和编辑发送E-mail，使所有读者受益。

Wrox支持过程只能直接支持与所出版图书有关的问题。关于正常支持范围以外的问题，参见http://p2p.wrox.com/forum的社区清单。

p2p.wrox.com

有关作者和相关讨论请加入P2P邮件清单。这个独特系统提供编程人员之间联系的邮件清单、论坛和新闻组，是一对一E-mail支持系统的补充。请相信，你的询问会得到许多Wrox作者和行业专家的检查。在p2p.wrox.com站点，可以找到不同的帮助清单，不仅在阅读本书时，而且在开发自己的应用程序时有用。

要预订邮件清单，步骤如下：

1. 访问http://p2p.wrox.com/。
2. 从左边菜单栏中选择相应类别。
3. 单击要加入的邮件清单。
4. 按照指令预订和填入E-mail地址与口令。

5. 答复收到的确认E-mail。
6. 用预订管理器加入多个清单和设置E-mail选项。

这些系统提供最佳支持

可以选择加入邮件清单，也可以接收每周摘要。如果没有接收邮件清单的时间或设备，则可以搜索档案。垃圾邮件会删除，Lyris系统会保护你的E-mail地址。加入与退出邮件清单的查询和关于邮件清单的其他查询可以发到listsupport@wrox.com。

目 录

第1章 J2EE设计模式

即使利用最先进的软件平台Java 2 Platform Enterprise Edition（J2EE），开发企业应用程序仍然是个难题。J2EE通过API提供技术与服务的高层抽象，使企业开发得到简化。但是，仅仅知道J2EE API是不够的。要设计良好的体系结构，得到高质量的应用程序，就要知道何时如何正确使用J2EE API。本书介绍如何正确使用J2EE API、技术与服务。

由于新技术方面的经验缺乏，我们通常要自己猜测如何正确使用。通常很难第一步就走对，而是要不断试验，直到找出最佳方法。最佳方法显然是从实践中得到的，不是发明出来的，而是发现和改善而成的。

工程学科中的一大原则是总结经验和利用实践证明有效的方案，企业应用程序也是这样。经验有助于更快更顺利地建立良好方案，从而节省成本，提高质量。惟一的问题是需要获得经验，但这个经验可以是别人的间接经验，而不一定非要是自己的直接经验。本书帮助读者节省自己的时间，介绍如何根据许多开发人员的间接经验进行合理设计。

但是，别人的经验如何描述呢？多年来，模式已经成为收集、规范和分析某些情境中常见问题的有效方法。本书主要介绍J2EE设计模式，不仅介绍J2EE企业开发中的常见问题及其解决方案，而且介绍其如何运用到实际项目中。

本章概述模式，并特别介绍设计模式与J2EE，同时介绍一些准则，如怎样有效利用设计模式。本章介绍的内容包括：

- 模式的演变
- 何谓设计模式
- 标识模式
- 设计模式如何解决设计问题
- 选择设计模式
- 使用设计模式
- 反模式
- J2EE与设计模式
- J2EE特定模式与J2EE情境中的模式
- J2EE模式的问题域

模式的演变

设计模式是从建筑与民用工程开始的。20世纪70年代，建筑学是需要大量经验的学科。Chris-topher Alexander等的三本著作震惊了整个行业，使不需要太多专门知识与经验的人也可以使用建筑学。他们标识有效结构之间的相似性，确定共同原则，作为常见设计问题的方案，并将其命名为建筑学中的方案“模式”。

但是，发布253个模式之后，Christopher Alexander仍然没能使建筑学成为机械化过程。那么，这些建筑模式有什么重要性呢？模式的目标并不是使建筑学成为机械化过程，而是为了普及常见建筑问题的解决方案，使经验较少的建筑师能更高效地进行设计，这是建筑模式的主要目标。

建筑模式对其他工程学科产生了巨大的影响。显然，每个成熟的工程学科都应建立常见问题的解决方案，软件工程也不例外，可以用模式成功地描述常见软件问题的解决方案。模式不仅提供良好方案，而且还可以帮助人们进行沟通，帮助他们分析工作和寻找原因。

软件工程中的模式

软件工程中的模式是什么呢？没有一个公认的定义。

简单地说，模式就是情境中一个问题经过证实的一个方案。

但是，这个简单定义可能造成对模式的误解。Christopher Alexander写道：

“每个模式描述一个在我们的环境中不断重复出现的问题，然后描述这个问题的方案核心，使你可以多次使用同一方案，而不必进行重复工作。”

近年来关于软件工程中的模式的最有影响的出版物是《Design Patterns: Elements of Reusable Object-Oriented Software》，作者Erich Gamma、Richard Helm、Ralph Johnson与John Vlissides称为四人帮；Addison-Wesley出版公司出版（ISBN: 0-201-63361-2）。其定义模式如下：

“模式是三段值，表示特写情境、问题与方案之间的关系。”

根据这个定义，模式是多种情形中可以使用的问题解。

Richard Gabriel提供了一个有趣的定义：

“每个模式是三段值，表示情境，这个情境中重复出现的问题及解决这些问题的一定软件配置之间的关系。”

模式是从经验中总结出来的，基于经过证实的方案。模式只有在实际系统中多次得到验证之后才能成为模式。因此，模式一方面促进复用，一方面又防止重复劳动，最终使我们能更快更高效地工作。

模式还增加表达能力，改进建筑师与设计人员之间的通信，使我们可以按前所未有的方式考虑常见结构性方案。模式鼓励通过结合解决大问题。

由此可见，模式没有什么新东西。有经验的设计人员可以阅读模式和发现过去的处理方法。众所周知，专家并不从低级结构考虑问题，而是建立高级抽象。但模式则鼓励人们标识与记录这些高级抽象。标识模式的动机包括：

- 寻找模式靠经验而不是靠想像。想像会引入风险，因为新的方法与技术要经过验证与测试之后才能证明有用。实际中，成功通常比想像更重要。因此，新的模式要经过试验才能评估其价值。好的模式是从实践中总结出来的。

- 模式能改进开发人员之间的通信，使他们可以更快地学习，从而开发更好的软件。为了达到高效通信，我们要用某种方式表示模式，最好使用标准格式。稍后将介绍表示模式的格式。
- 模式可以按照质和量验证知识，从而评估其价值。这样可以表示与理解模式及所建体系结构的利弊。

模式几乎无所不在。多年来，出现了多种不同的软件模式，包括：

- 设计模式，包括软件设计，可能是最重要的模式。通常是面向对象的，包括体系结构（系统设计）、设计（组件交互）和编程（特定语言技术）。本书主要介绍设计模式。
- 分析模式，描述可复用分析模型，在域分析中非常有用，涉及各种域，包括交易、度量、会计和组织关系。要了解分析模式的更多信息，见《Analysis Patterns: Reusable Object Models》一书，Addison-Wesley出版（ISBN 0-201-89542-0）。
- 过程模式，描述软件过程设计。具体地说，它们描述开发软件成功而经过证实的方法与活动。详见剑桥大学出版社的《Process Patterns: Building Large-Scale Systems Using Object Technology》（ISBN 0-521-64568-9）与《More Process Patterns: Delivering Large-Scale Systems Using Object Technology》（ISBN 0-521-65262-6）。
- 组织模式，描述组织与项目的结构和实践。详见http://i44pc48.info.uni-karlsruhe.de/cgi-bin/OrgPatterns。
- 实现模式，描述实现概念。详见《Essential Java Style: Patterns for Implementation》一书，Prentice Hall出版（ISBN 0-13-085086-1）。
- 其他域特定模式。

模式还可以根据包括的问题分为：

- 创建性模式
- 结构性模式
- 行为性模式

可以看出，模式有几个抽象层，可以按不同方式分类。本书主要介绍J2EE设计模式。首先介绍设计模式。

何谓设计模式

设计模式是情境中标准设计问题的重复性解决方案。设计问题必须进一步调查之后才能解决。问题通常发生在一定的环境或情形中，称为情境。解决方案是这些问题的答案，帮助我们在一定情境中解决这些问题。

例如，假设有一个小房间，问题是把门放在哪里更方便。经过不同测试之后，我们可能发现应把门放在房间的东南角。这样就找到了特定问题的解。同样，软件设计中也可以找到特定问题的解。

设计问题的解是否都是设计模式呢？不一定。设计模式只是适用于不同情境的解，即可以重复采用，在不同情境中解决同一问题。

回到房门的问题，这个解还不能作为模式，因为它只限于这个特定情形。但如果能对所有小房间找到一般解，则可以称为模式。Alexander就是这么干的。他发现，房门不能放在墙上任何地方。此外，房间的成功很大程度上取决于门的位置。因此，他建议除了很大的房间之外，房门应尽量靠近墙角。他把这个模式称为角门（Corner Doors）模式。

虽然这是一个简单概念，但定义设计模式并不容易。因此，人们提出了五花八门的定义。为了深入了解设计模式，下面再进一步解释。

设计模式针对软件设计，系统化地命名、解释和求值重要软件设计。设计模式使成功地经过证明的设计与体系结构更容易复用，使新系统开发人员能更方便地采用设计模式，特别是经验较少的开发人员。设计模式能帮助我们选择不同设计。本书主要介绍设计模式，因此此后“模式”一词特指设计模式。

假设要开发Customer实体Bean，并把客户数据提供给客户机。自然方法是用细粒方法定义远程接口（如get/setName()、get/setAddres()），让远程客户机直接访问这个实体Bean。要确定这个解不可伸缩，因此不适合实际应用，需要实现组件与客户机，进行部署和测试，需要进行大量工作。记住，实际应用程序中通常要面对多个组件，而不只是Customer。

知道Value Object与Session Façade之类的设计模式之后，就很容易看出EJB的远程接口应是粗粒的。Value Object（数值对象）模式显示了如何使接口变成粗粒，同时以巧妙的方式传输所有必要的数据，避免多次远程方法调用。Session Façade模式显示不能直接访问实体bean，而应开发会话bean，作为门户。如果能对实体bean增加一个本地接口，在同一容器中部署两个EJB，则可以进一步提高性能。

如果你不熟悉Value Object与Session Façade模式，则不容易理解上段的内容。别担心，这些模式都将在本书稍后详细介绍。但可以注意，这两个模式都适用于某种情境。所有模式都适用于某种情境，是一组矛盾的平衡。描述模式时，我们要明确定义所有这些项目。根据前面提到的四位专家对设计模式的描述，模式具有四大要素：

- 模式名，标识模式，增加表达性。
- 问题，描述何时应用这个模式，解释问题与情境。
- 解，不描述具体设计与实现，而是描述模式模板，可以在不同情境中使用。
- 结果，是采用一个模式的结果与取舍。结果对评估不同方法和进行决策至关重要。

设计模式描述如何在特定情境中解决一般设计问题。

设计模式对常见设计结构的关键方面进行命名、抽象和标识。标识方案参与者，他们的角色与责任及合作方式。大多数情况下，这些参与者是对象与组件。每个设计模式针对特定问题。每个模式还描述其结果和取舍。

相关模式交织在一起，构成模式语言。模式语言不是正式语言，而是一组相关模式的集合。模式和模式语言有助于更好、更快、更有效地学习、通信和解决问题。本书主要介绍J2EE的模式语言，但介绍J2EE设计模式及其应用之前，先要介绍如何标识模式。

标识模式

标识模式要靠经验。每个有经验的建筑师和设计人员能够看到，在大部分项目中会遇到类似问题，这些问题的解也是相似的。当然，实际解不一定是相同的。但是，所有解具有相同的基本概念。从抽象角度看，可以说所有这些解与问题表示具有共同抽象解的共同抽象问题。

了解这一点是标识模式的基础。我们应回顾过去的项目，标识处理过的问题。对每个问题，应标识其特征，还应记录这些问题的解。然后收集类似问题与解，确定其相似性。

类似解很可能表示一个模式。前面曾介绍过Alexander的角门模式，Alexander通过大量房间中门的布置位置找到这个模式。所有把门放在角上的房间都采用这种模式。前面介绍的Value Object与Session Façade模式也是这样，也是通过分析多个应用程序而得到的。

但是，一组解只有经过验证之后才能成为模式。由于Alexander验证把门放在角上是有用的，我们也要验证所找到的模式。在软件工程中验证复杂模式可能比验证角门模式之类的简单建筑模式更复杂。

因此，我们不是直接标识模式，而是标识模式候选者。模式候选者是独立方案，与外部的连接越少越好。记住，模式是为了复用。我们要验证模式候选者，通常使用三规则，即每个模式候选者至少要在三个不同系统中证明之后才能成为真正的模式。这个规则不太精确，因此本书作者建议模式候选者应尽量多次验证，使模式更有实用价值。

没有充分验证的模式可能带来大量危害。为什么呢？因为大多数建筑师、设计人员和开发人员并不标识模式，可能没有时间，也可能没有这方面的经验。但他们通常都学习模式和用其改进解决方案。无法达到目标的模式可能对大量应用程序造成伤害，从而给模式带来不好的名声。

另一个重要问题是模式应采用哪一级抽象，包括多少实现细节。J2EE模式社区中普遍接受的概念是模式应采用高层抽象，但每个模式应包括解的细节。这些解的细节通常比模式本身采用更低层抽象，称为策略。显然，模式与策略之间没有明显的分界。

由于本书介绍模式，因此会通过这些策略显示不同情境中如何采用模式。

在Prentice Hall PTR出版的《Core J2EE Patterns》一书（ISBN 0-13-064884-1）中，作者定义了模式与策略的下列差别：

- 模式应采用更高层抽象，因此应提出最佳实现策略。
- 开发人员可以通过策略扩展使用模式，寻找实现模式的新方法。
- 模式与策略名称改进通信。

标识模式之后，要表示模式。下节介绍如何表示设计模式。

表示设计模式

表示模式和定义模式一样难，仅仅靠统一建模语言（UML）之类的简单图形表示方法是不够的。为了使模式真正促进复用，就要表示意图、动机、决策、后果，等等。要成功运用

模式，就应提供具体例子。

近年来，人们找到了多种模式。标识这些模式的作者将其发表到模式类别中。也许最重要的模式类别是前面提到的四位专家在设计模式中提供的23种模式。对J2EE开发人员，模式类别是个社区过程，见Java Developer Connection站点http://java.sun.com/。J2EE开发人员的另一重要模式类别见http://www.TheServerSide.com/。

模式可以用正式与非正式方式表示。如今大多数模式类别用非正式方式表示模式。但问题是他们不用相同格式，而采用稍有不同的格式。模式表示的主要差别源于不同类型模式解决的设计问题并不相似。例如，前面提到的四位专家的模式解决面向对象设计问题，而核心J2EE模式解决建立J2EE应用程序时遇到的问题。

非正式方式表示模式时通常使用几段，有些是大多数表示中都有的，有些是不同的。大多数非正式方式表示基于前面提到的四位专家的著作中使用的表示。前面提到的四位专家的著作中使用的表示采用下列模板：

- **Pattern name and classification（模式名与类别）**
  每个模式应有惟一名称，使模式便于口头表达。
- **Intent（意图）**
  简要描述模式的作用、理由与意图、解决的设计问题。
- **Also Known As（别名）**
  提供模式的别名名单。
- **Motivation（动机）**
  描述问题内容及如何用这个模式解决这个问题。
- **Applicability（适用性）**
  描述设计模式的适用情形，不好的设计例子及如何识别。
- **Structure（结构）**
  提供图形表示，可以使用UML图形。
- **Participants（参与者）**
  显示参与这个设计模式的类、对象和组件，及各处的责任。
- **Collaborations（协作）**
  描述参与者如何协作。
- **Consequences（后果）**
  列出模式如何达到目标，使用模式的结果与取舍。
- **Implementation（实现）**
  实现模式的提示、技术与策略，以及特定语言问题和注意事项。
- **Sample code（样本代码）**
  显示如何实现模式样本代码。
- **Known uses（已知用途）**
  实际系统中如何使用这个模式。
- **Related patterns（相关模式）**
  列出相关模式及主要差别。