TURING 图灵程序设计丛书 Java 系列

Apress®

Pro Java EE Spring Patterns

Best Practices and Design Strategies Implementing Java EE Patterns with the Spring Framework

Java EE设计模式

Spring企业级开发最佳实践

[印] Dhrubojyoti Kayal 著
张平 龚波 李平芳 等译

- 在Spring框架下实现Java EE设计模式
- 剖析各个层里常用的21种模式
- 理论与实践完美结合，相得益彰

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书

-38

Pro Java EE Spring Patterns

Best Practices and Design Strategies Implementing Java EE Patterns with the Spring Framework

Java EE设计模式

Spring企业级开发最佳实践

[印] Dhrubojyoti Kayal 著
张平 龚波 李平劳 等译

人民邮电出版社
北 京

图书在版编目（CIP）数据

Java EE设计模式 : Spring企业级开发最佳实践 / (印) 凯耶尔 (Kayal,D.) 著 ; 张平等译. -- 北京 : 人民邮电出版社, 2010.2
（图灵程序设计丛书）
书名原文: Pro Java EE Spring Patterns:Best Practices and Design Strategies Implementing Java EE Patterns with the Spring Framework
ISBN 978-7-115-22129-2

Ⅰ. ①J… Ⅱ. ①凯… ②张… Ⅲ. ①JAVA语言－程序设计 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2010)第007530号

内 容 提 要

本书结合 Spring 框架讲解了 Java EE 设计模式，主要介绍了 Java EE 应用程序设计和 Spring 框架的基础知识，描述了表现层、业务层和集成层中使用的设计模式，提供了每个模式的实现细节并分析了其优缺点，最后运用书中所讲的内容示范了开发订单管理系统的过程。

本书主要适合 Java EE 应用程序设计人员和架构师使用。

图灵程序设计丛书

Java EE设计模式——Spring企业级开发最佳实践

◆ 著 [印] Dhrubojyoti Kayal
译 张 平 龚 波 李平芳 等
责任编辑 王军花

◆ 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
北京艺辉印刷有限公司印刷

◆ 开本：800×1000 1/16
印张：15
字数：355千字 2010年2月第1版
印数：1－3 000册 2010年2月北京第1次印刷

著作权合同登记号 图字：01-2009-2892号

ISBN 978-7-115-22129-2

定价：45.00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

感谢我的父母和妻子!

译 者 序

《设计模式》的作者John Vlissides认为："Java世界到处充满代码库、工具和规范。现在迫切需要的是把这些东西归纳为能够解决实际问题的技术。模式就是J2EE软件开发的智能发动机。"

本书集中介绍企业级模式、最佳实践和设计策略，并提供使用Java EE关键技术（比如JSP、servlet、EJB和JMS API等）的解决方案。细读本书，你能了解企业级Java/Java EE应用程序设计模式，学习使用流行的Spring框架来简化企业Java设计，掌握表现层、业务层和集成层的设计模式和最佳实践，包括横切设计模式、AOP等。

本书所面向的读者包括打算或者正在使用Spring框架的Java EE应用程序架构师、设计者和开发者。阅读本书需要具备Java EE设计模式、Spring框架以及Eclipse IDE等基本知识。

本书主要由龚波、张平主持翻译，龚波负责最后的统稿。其他参与本书翻译和审校工作的人员还有徐雅丽、李平芳、李志、刘刚、任志宏、王强等。感谢大家的辛勤工作和专业精神。同时，还要感谢出版社编辑老师的辛勤工作！

虽然在翻译过程中竭尽所能，但不可否认本书中肯定存在翻译或者理解不当的地方，希望读者朋友能够给予善意的批评和指正！

致　谢

我想借此机会感谢那些为本书贡献了思想和灵感以及付出辛勤劳动的人们。首先，要感谢Steve Anglin给我撰写此书的机会。在2007年9月动手写这本书时，我们的想法截然不同。后来，是Steve提出了整合Spring框架和Java EE设计模式的思想。

非常感谢Prosenjit Bhattacharyya和Tom Welsh在技术评审上花费了大量的时间。从大学时代起，Prosenjit就是我的好朋友，本书每一章的结构他都给出了反馈意见（尤其是第7章）。从Tom那里我学到了大量写作知识，比如如何以简洁明了的方式正确讲述主题。

在此，还要特别感谢Kylie Johnston。Kylie是我见过的最具耐心和合作精神的项目经理。必须承认，如果没有她的帮助，这本书的出版可能就遥遥无期了。写这本书时，我拖期了好几次。正是Kylie一次又一次地提醒我最后期限，又没有放松质量上的要求。还必须感谢Kim Wimpsett、Laura Cheu和Elizabeth Berry为本书出版付出的辛勤劳动。

还要感谢我在Cognizant公司工作时的同事——Suman Ray和Somnath Chakraborty，正是他们的引导和鼓励，让我走上了技术职业生涯。本书第7章中讲述的设计指令理念是Somnath于2005年提出的，并获得了巨大的成功。

前　言

本书将Java EE设计模式和Spring框架融合在了一起。对于任何Java EE应用程序的设计和架构工作来说，Java EE设计模式都有极大的参考价值。另一方面，Spring框架是Java EE框架的事实标准。Spring具有简单的编程模型，并强调对象设计的最佳实践，有助于改进和增强Java EE平台的适用性。

很长时间以来，我始终使用Spring框架和设计模式来构建Java EE应用程序。本书致力于归纳在Spring框架中常用的设计策略（符合最新的Java EE 5规范）。我坚信，本书可供那些有兴趣使用Java EE和Spring框架构建企业级应用程序的设计人员和开发人员参考。

本书的读者对象

本书主要适合Java EE应用程序设计人员和架构师使用，也非常适合熟悉Java EE设计模式和Spring框架的开发人员使用。

本书的组织结构

本书的组织结构非常简单。第1章首先介绍企业级应用程序架构中所用的基本概念，分析了分布式计算的各种架构风格，还介绍了使用UML工具进行应用程序的可视化设计。

第2章介绍了Spring框架及其在构建企业级Java应用程序中的作用，重点介绍了后续4章会使用的设计模式模板。第3章解释了表现层的设计问题，并提出基于Spring MVC框架的解决方案。第4章详细阐述业务层的设计模式，并介绍Spring对简化EJB开发的支持。

第5章讨论集成层的设计模式。第6章讨论通常容易被忽视的安全和事务设计策略。最后，第7章使用前几章提到的概念展示了一个订单管理系统的设计开发过程。

预备知识

本书假设读者熟悉Java EE设计模式、Spring框架以及Eclipse IDE。

源代码下载

访问http://www.apress.com，找到本书的主页，即可看到源代码下载链接。欢迎随时访问Apress

的Web站点，并下载本书的所有源代码[①]。你也可以获取Apress提供的勘误表，并查找相关信息。

联系作者

读者可以通过dhrubo.kayal @ gmail.com随时与作者联系。

① 本书源代码也可以从图灵公司的网站www.turingbook.com的本书主页上下载。——编者注

书号：978-7-115-20890-3

Spring攻略

- **Spring专家力作**
- **理论与实践完美结合**
- **问题描述→解决方案→实现方法**

内容简介

本书是目前国外人气最高、口碑最好的一本 Spring 图书。本书内容翔实，示例生动丰富，代码实用，可操作性强。它不仅涵盖了 Spring 2.5 从基础概念到高级应用的所有主题，而且深入浅出地介绍了几种常见的 Spring 项目，其参考价值不言而喻。

书中采用了“问题描述→解决方案→实现方法”的方式，读者可以轻松地查找特定问题的解决方案，事半功倍。通过本书，可以迅速掌握 Spring 来构建强大的企业级 Java 应用程序，成为 Spring 高手。

媒体评论

“……此书的内容如此朴实无华，但正是我苦苦寻觅的，这让我非常震惊！”

——Damodar Chetty，Software Engineering Solutions, Inc.

“我很少发表评论，但这次是例外。这是迄今为止最好的图书，你一定会手不释卷。此书可读性极好，内容结构严谨有序。我真的很惊奇，它怎么能够如此详细。”

——Amazon.com 评论

书号：978-7-115-21204-7

Spring高级程序设计

- **Spring框架创始人倾情推荐的权威开发指南**
- **全面揭示Spring框架关键技术**
- **深入了解Spring内部工作机制**

内容简介

作为最强大、应用最广泛的企业级 Java 开发框架，Spring 因其强大的适应性和可扩展性而适用于各种企业级系统。本书是由资深 Spring 开发专家编写，囊括了 Spring 开发人员需要了解的精炼要点和复杂主题。基于目前企业中应用最广泛的 Spring 2.5 版本，不但全面介绍了 Spring 开发框架的关键技术和模块，并且还介绍了 AJAX、Web 工作流、动态语言等主流技术。

本书适合所有 Java 开发人员，特别是企业级 Java 开发人员阅读参考。是一本由 Spring 框架核心开发人员为读者奉献的权威开发指南，将带给你设计和构建高效、可扩展的 Spring 应用的丰富知识和经验。

媒体评论

Spring Framework 2.5 的发布反映了 Spring 框架和企业 Java 框架的最新进展。任何勤勉的 Java 开发者都应该阅读这本开发指南。

——*Pro Spring* 一书作者 Rob Harrop

目　录

第 1 章　企业级 Java 应用程序架构和设计简介 ····· 1

1.1　分布式计算的发展历程 ····· 1
1.1.1　单层架构 ····· 2
1.1.2　两层架构 ····· 2
1.1.3　三层架构 ····· 2
1.1.4　多层架构 ····· 4
1.1.5　Java EE架构 ····· 4
1.2　Java EE应用程序设计 ····· 8
1.3　Java EE设计模式目录 ····· 9
1.4　使用UML描述Java EE架构和设计 ····· 10
1.4.1　类图 ····· 10
1.4.2　序列图 ····· 12
1.5　小结 ····· 13

第 2 章　使用 Spring 框架简化企业级 Java 应用程序 ····· 14

2.1　什么是Spring ····· 14
2.2　为什么Spring很重要 ····· 14
2.3　Spring框架的组成部分 ····· 16
2.3.1　Spring Core ····· 16
2.3.2　Spring AOP ····· 22
2.3.3　Spring DAO ····· 23
2.3.4　Spring ORM ····· 23
2.3.5　JEE ····· 23
2.3.6　Web MVC ····· 23
2.4　使用Spring构建分层应用程序 ····· 23
2.4.1　表现层 ····· 24
2.4.2　业务层 ····· 25
2.4.3　集成层 ····· 25
2.5　Spring Java设计模式讲解模板 ····· 26
2.5.1　名称 ····· 26
2.5.2　问题描述 ····· 26
2.5.3　模式目的 ····· 26
2.5.4　解决方案 ····· 26
2.5.5　模式评价 ····· 26
2.6　小结 ····· 26

第 3 章　表现层设计模式 ····· 27

3.1　前端控制器 ····· 28
3.1.1　问题描述 ····· 28
3.1.2　模式目的 ····· 30
3.1.3　解决方案 ····· 30
3.1.4　模式评价 ····· 33
3.2　应用程序控制器 ····· 33
3.2.1　问题描述 ····· 33
3.2.2　模式目的 ····· 34
3.2.3　解决方案 ····· 34
3.2.4　模式评价 ····· 46
3.3　页面控制器 ····· 47
3.3.1　问题描述 ····· 47
3.3.2　模式目的 ····· 47
3.3.3　解决方案 ····· 47
3.3.4　模式评价 ····· 63
3.4　上下文对象模式 ····· 64
3.4.1　问题描述 ····· 64
3.4.2　模式目的 ····· 64
3.4.3　解决方案 ····· 64
3.4.4　模式评价 ····· 70
3.5　拦截过滤器模式 ····· 70
3.5.1　问题描述 ····· 70
3.5.2　模式目的 ····· 70

3.5.3 解决方案 …… 71
3.5.4 模式评价 …… 76
3.6 视图助手模式 …… 76
3.6.1 问题描述 …… 76
3.6.2 模式目的 …… 76
3.6.3 解决方案 …… 77
3.6.4 模式评价 …… 84
3.7 组合视图模式 …… 85
3.7.1 问题描述 …… 85
3.7.2 模式目的 …… 85
3.7.3 解决方案 …… 85
3.7.4 模式评价 …… 89
3.8 分发者视图模式 …… 89
3.8.1 问题描述 …… 89
3.8.2 模式目的 …… 89
3.8.3 解决方案 …… 90
3.8.4 模式评价 …… 94
3.9 服务到工作者模式 …… 94
3.9.1 问题描述 …… 94
3.9.2 模式目的 …… 94
3.9.3 解决方案 …… 95
3.9.4 模式评价 …… 95
3.10 小结 …… 96
第 4 章 业务层设计模式 …… 97
4.1 服务定位器模式 …… 97
4.1.1 问题描述 …… 97
4.1.2 模式目的 …… 100
4.1.3 解决方案 …… 100
4.1.4 模式评价 …… 109
4.2 业务代理模式 …… 109
4.2.1 问题描述 …… 109
4.2.2 模式目的 …… 109
4.2.3 解决方案 …… 109
4.2.4 模式评价 …… 111
4.3 会话外观模式 …… 112
4.3.1 问题描述 …… 112
4.3.2 模式目的 …… 112
4.3.3 解决方案 …… 112
4.3.4 模式评价 …… 116
4.4 应用程序服务模式 …… 117
4.4.1 问题描述 …… 117
4.4.2 模式目的 …… 117
4.4.3 解决方案 …… 118
4.4.4 模式评价 …… 120
4.5 业务接口模式 …… 121
4.5.1 问题描述 …… 121
4.5.2 模式目的 …… 121
4.5.3 解决方案 …… 121
4.5.4 模式评价 …… 127
4.6 小结 …… 127
第 5 章 集成层设计模式 …… 128
5.1 数据访问对象模式 …… 128
5.1.1 问题描述 …… 128
5.1.2 模式目的 …… 131
5.1.3 解决方案 …… 131
5.1.4 模式评价 …… 140
5.2 过程访问对象模式 …… 140
5.2.1 问题描述 …… 140
5.2.2 模式目的 …… 140
5.2.3 解决方案 …… 140
5.2.4 模式评价 …… 143
5.3 服务触发器模式 …… 143
5.3.1 问题描述 …… 143
5.3.2 模式目的 …… 144
5.3.3 解决方案 …… 144
5.3.4 模式评价 …… 151
5.4 Web服务代理模式 …… 151
5.4.1 问题描述 …… 151
5.4.2 模式目的 …… 151
5.4.3 解决方案 …… 152
5.4.4 模式评价 …… 161
5.5 小结 …… 161
第 6 章 横切设计模式 …… 162
6.1 验证和授权实施者模式 …… 163
6.1.1 问题描述 …… 163
6.1.2 模式目的 …… 164
6.1.3 解决方案 …… 164

6.1.4 模式评价 …… 182
6.2 审核拦截器模式 …… 182
6.2.1 问题描述 …… 182
6.2.2 模式目的 …… 182
6.2.3 解决方案 …… 183
6.2.4 模式评价 …… 189
6.3 域服务所有者事务模式 …… 189
6.3.1 问题描述 …… 189
6.3.2 模式目的 …… 189
6.3.3 解决方案 …… 190
6.3.4 模式评价 …… 197
6.4 小结 …… 197
第 7 章 案例研究：构建订单管理系统 …… 198
7.1 需求 …… 198
7.1.1 用户故事卡：用户登录 …… 199
7.1.2 用户故事卡：查询服务 …… 199
7.1.3 用户故事卡：保存订单 …… 199
7.2 迭代规划 …… 199
7.3 架构 …… 200
7.3.1 表现层 …… 200
7.3.2 业务层 …… 201
7.3.3 集成层 …… 202
7.4 设计 …… 202
7.5 安全机制 …… 203
7.5.1 问题描述 …… 203
7.5.2 模式目的 …… 203
7.5.3 解决方案 …… 203
7.6 JSP …… 203
7.6.1 问题描述 …… 203
7.6.2 模式目的 …… 204
7.6.3 解决方案 …… 204
7.7 页面控制器 …… 204
7.7.1 问题描述 …… 204
7.7.2 模式目的 …… 204
7.7.3 解决方案 …… 204
7.8 开发 …… 205
7.8.1 创建工作区 …… 206
7.8.2 创建项目 …… 207
7.8.3 添加依赖关系 …… 208
7.8.4 构建项目 …… 210
7.8.5 部署项目 …… 219
7.9 小结 …… 227

第1章 企业级Java应用程序架构和设计简介

长期以来，Java EE（Java企业版）已成为各行业（银行、保险、零售、医疗、旅游以及电信等）开发和部署企业级业务应用程序的首选平台。这是由于Java EE提供了一个基于标准的平台，可用来构建强壮和高扩展性的分布式应用程序，以支持从银行核心业务到航空订票引擎在内的所有业务。但是，开发成功的Java EE应用程序却是一项非常艰巨的任务。首先，Java EE平台提供的丰富选项就大得惊人。数量繁多的框架、实用工具库、集成开发环境（IDE）以及各种工具，使得开发工作更具挑战性。因此，在开发基于Java EE的软件时，选择合适的技术是至关重要的。选择使用具有良好架构和设计的技术，才有可能构建易于维护、复用和扩展的应用程序。

本章简要介绍Java EE应用程序架构和设计的基本内容，它们是整个应用程序开发的基石。

首先，简要回顾分布式计算和多层应用程序架构的发展历程，随后说明Java EE平台架构是如何解决开发分布式应用时所遇到的相关难题的。在这个过程中，读者也将了解MVC（Model-View-Controller，模型–视图–控制器）架构的基本原理。然后，将MVC原理应用于Java EE平台的开发，以构建多层的Java EE应用程序架构。

讨论完应用程序架构之后，本章将侧重介绍基于面向对象原理的Java EE应用程序设计，还将解释如何使用设计模式和最佳实践来简化应用程序设计。随后，介绍Sun公司Java Blueprints（Java蓝图）中所记载的Java EE设计模式，然后详细介绍Deepak Alur等人所著的*Core J2EE Design Pattern*（Prentice Hall，2003）中的设计模式。最后，本章将介绍统一建模语言（UML）及其在Java EE设计和架构的文档可视化方面的作用。

1.1 分布式计算的发展历程

在分布式计算中，应用程序被分割成同时在多台计算机上运行的若干个小应用程序。分布式计算也被称为网络计算（network computing），这是因为这些小应用程序一般采用基于TCP/IP或UDP的协议进行网络通信。这些更小规模的应用程序也被称作层（tier）。每个层都提供可供连接层或者客户层使用的独立的服务集合。这些层可以进一步划分为多个规模更小的层（layer），以提供更精细的功能。大多数应用程序都有3个层。

- 表现层（Presentation Layer）：用来处理用户界面相关处理操作。
- 业务层（Business Layer）：负责执行业务规则。在这个过程中，它也会与数据访问层进行交互。
- 数据访问层（Data Access Layer）：负责检索和处理存储在企业信息系统（Enterprise Information System，EIS）中的数据。

通过分析分布式应用程序架构的演进历史，我们可以更加清晰认识现代网络计算的现状。下面几节将结合适当的实例来介绍分布式架构的演进过程。

1.1.1　单层架构

单层架构（single-tier architecture）可追溯到连接哑终端的独立主机时代。整个应用程序由诸如用户界面、业务规则以及数据等逻辑层组成，它们都部署在同一台物理主机上。用户使用终端或者控制台与系统交互，终端或者控制台只具备非常有限的文本处理能力（参见图1-1）。

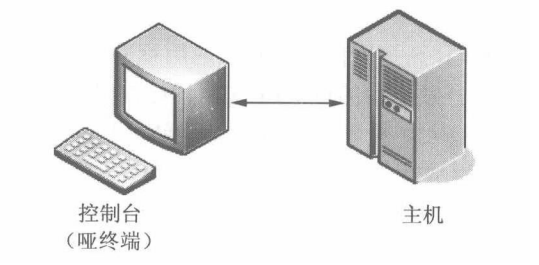

图1-1　单层架构

1.1.2　两层架构

20世纪80年代初期，PC（个人电脑）开始流行。这些PC的价格相对低廉，相对于哑终端来说，具有更加强大的数据处理能力。这为真正的分布式计算［即客户-服务器（client-server）计算］提供了基础。这时候，客户端或者PC只运行用户界面程序。它还支持图形化用户界面（Graphical User Interface，GUI），允许用户输入数据，并与主机服务器进行交互。这时候，主机服务器仅驻留业务规则和数据。数据输入完成后，GUI应用程序可以在客户端上执行数据验证，然后将数据发送至服务器，以执行业务逻辑。Oracle的基于Forms的应用程序就是一个很好的两层架构实例。在PC上加载以窗体形式提供的GUI，而业务逻辑（编码为存储过程）和数据则仍然驻留在Oracle数据库服务器上。

这时还存在另外一种两层架构，在这种架构中，不仅UI，甚至连业务逻辑也驻留在客户层。这种应用程序通常连接到数据库服务器，以执行各种不同的查询。由于客户层部署了大量可执行代码，所以这种类型的客户端被称作胖客户端（thick client或者fat client），见图1-2。

1.1.3　三层架构

两层胖客户端应用程序开发起来非常容易，但当需要进行软件升级时，由于涉及用户界面或者业务逻辑的变更，所以必须更新所有的客户端。20世纪90年代中期，硬件成本急剧下降，而CPU

处理能力却急剧提升，加之快速增长的因特网以及迅猛发展的基于Web的应用程序开发，这就催生了三层架构。

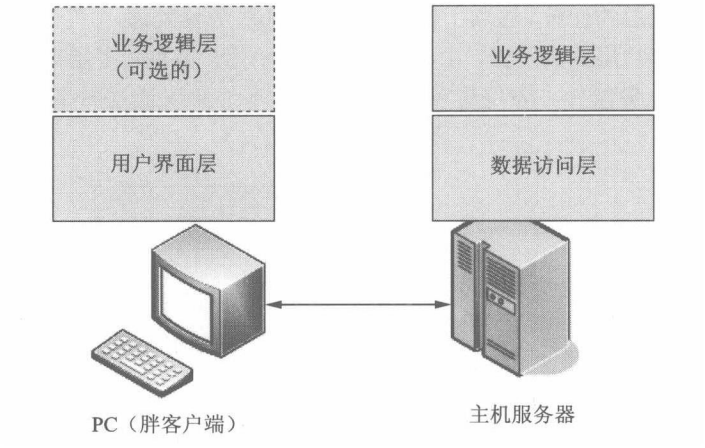

图1-2 两层架构

在三层架构中，客户端PC只需要部署瘦客户端软件（比如浏览器）以显示从服务器所返回的数据内容。服务器驻留表现逻辑、业务逻辑和数据访问逻辑。应用程序数据来自于企业信息系统，如关系型数据库。在这样的系统中，可远程访问业务逻辑，因此就有可能通过Java控制台应用程序来支持独立的客户端。业务层通常通过数据访问层和信息系统进行交互。由于整个应用程序都部署在服务器上，所以这种服务器也称作*应用程序服务器*（application server）或*中间件*（middleware）（参见图1-3）。

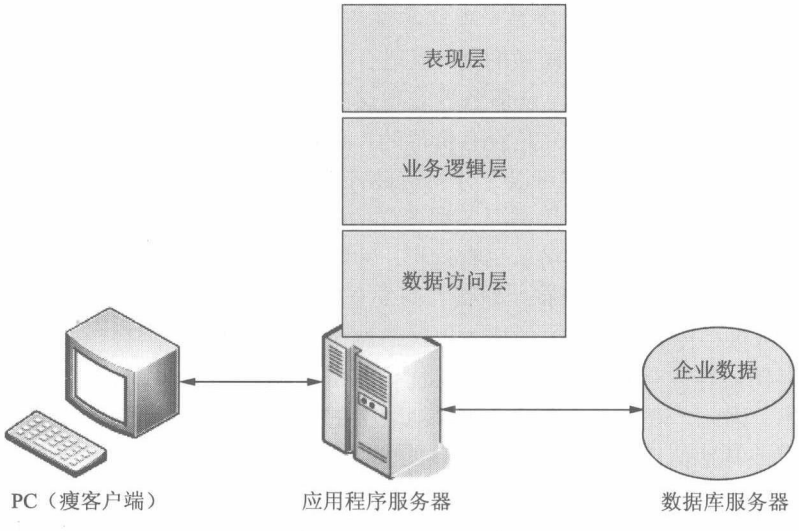

图1-3 三层应用程序

1.1.4　多层架构

随着因特网带宽的快速增长，世界各地的企业都提供基于Web的服务。这样，应用程序服务器就不再负责处理表现层的任务。这项工作由专用的Web服务器处理，由它生成需要显示的内容。生成的内容将传递给客户层的浏览器，由浏览器负责显示用户界面。多层架构的应用程序服务器上驻留可远程访问的业务组件。表现层Web服务器使用本地协议通过网络访问这些组件。图1-4说明了多层应用程序。

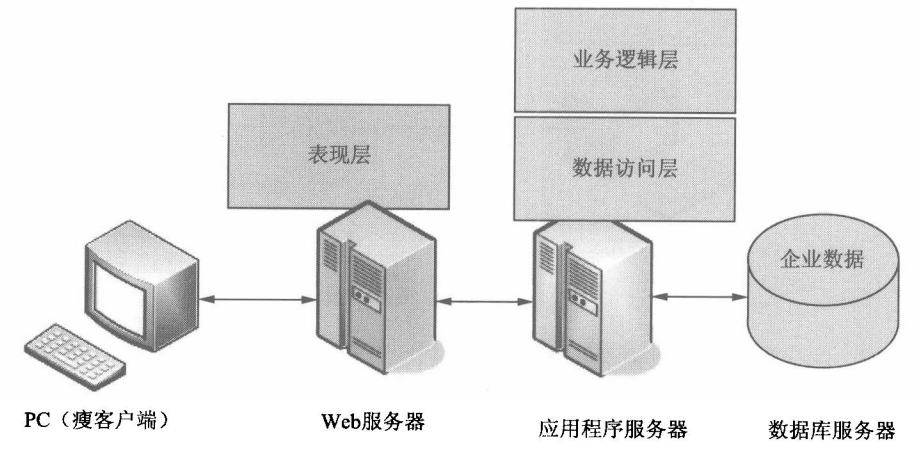

图1-4　多层应用程序

1.1.5　Java EE 架构

开发多层架构的分布式应用程序是一项非常复杂、极具挑战性的工作。为了更好地利用资源，人们把处理任务分摊到多个层。这样做的话，还适合为专家分配他最擅长实现和开发特定层的任务。例如，网页设计者更适合处理Web服务器表现层的事务，而数据库开发者则可集中开发存储过程和函数。但是，这些层相互孤立是没有价值的，必须将它们集成起来才能实现更大的企业级目标。这么做是必需的，因为这样可以充分利用最有效的协议，否则会导致性能急剧下降。

除集成之外，分布式应用程序还需要各种各样的服务。在和不同的信息系统交互时，这些服务必须能够创建、参与或者管理事务。只有这样，才能保证并发处理企业数据。因为是通过因特网访问多层应用程序的，所以必须有功能强大的安全服务的支持，阻止对应用程序的恶意访问。

现在，硬件（如CPU和存储器）的成本已经急剧下降，不过仍然存在一些限制，比如，处理器支持的最大存储器容量有限。因此，有必要优化使用系统资源。当前的分布式应用程序通常使用面向对象技术。因此，诸如对象缓存或对象池之类的服务是非常方便的。这些应用程序经常会与关系型数据库或其他信息系统（如面向消息的中间件）进行交互。不过，与这些系统建立连接的成本是非常大的，因为需要耗费大量的处理资源，并会严重降低应用程序性能。在这些情况下，连接池就显得特别有用，它不仅可以在相当大程度上改进应用程序性能，还能优化资源利用。