

宝典丛书

100万

Spring 2.0

宝典

结合示例程序进行讲解, 覆盖Spring绝大多数API

全面诠释Spring的AOP框架, 演示基于AOP框架的权限检查

结合实例讲解DAO等J2EE设计模式, 直接提升读者对架构设计的把握

代码注释极为详尽, 帮助读者快速掌握要领

最后配备了三个完整案例, 让读者从项目中体会Spring的奥妙

李刚 编著



电子工业出版社

Publishing House of Electronics Industry
<http://www.phei.com.cn>

宝典丛书

Spring 2.0 宝典

李刚 编著

电子工业出版社

Publishing House of Electronics Industry

北京·BEIJING

内 容 简 介

Spring 是目前最流行的 J2EE 框架, 在 J2EE 应用的各层都有其不俗的表现。Spring 提倡的“实用主义”法则大大简化了 J2EE 的开发。

本书由浅入深、全面地介绍了 Spring 的结构体系, 内容覆盖到 Spring 近 80% 的 API。全书分 21 章, 内容涵盖了 Spring 的核心机制、依赖注入、资源访问、AOP 框架、事务框架、整合 Hibernate、DAO 支持、JDBC 支持、MVC 框架、整合第三方表现层技术、整合第三方 MVC 框架、远程访问支持、EJB 访问和实现、Spring 对测试的简化、Spring 对 JMS 和 JavaMail 的支持等。本书的示例都经笔者精心挑选, 具有很强的针对性, 力求让读者可以明白 Spring 每个知识点。最后的两个综合案例, 采用最科学的轻量级 J2EE 结构, 涉及的框架有 Spring, Struts, WebWork2, Hibernate, FreeMarker, Velocity 等, 知识面相当全面, 具有很好的示范作用, 力争让读者感受到高质量 J2EE 应用的魅力。

本书适用于有较好的 Java 编程基础, 有一定的 J2EE 编程基础的用户。本书既可以作为 Spring 的学习指南, 也可作为实际开发人员的参考手册。

**未经许可, 不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有, 侵权必究。**

图书在版编目(CIP)数据

Spring 2.0 宝典 / 李刚编著. —北京: 电子工业出版社, 2006.10
(宝典丛书)
ISBN 7-121-03234-1

I.S... II.李... III.JAVA 语言—程序设计 IV.TP312

中国版本图书馆 CIP 数据核字 (2006) 第 115217 号

责任编辑: 张月萍 特约编辑: 明足群

印 刷: 北京东光印刷厂

装 订: 三河鹏成印业有限公司

出版发行: 电子工业出版社

北京市海淀区万寿路 173 信箱 邮编: 100036

开 本: 787 × 1092 1 / 16 印张: 46.25 字数: 1317 千字

印 次: 2006 年 10 月第 1 次印刷

定 价: 79.00 元 (含光盘 1 张)

凡所购买电子工业出版社图书有缺损问题, 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; 邮购电话: (010) 88254888。

质量投诉请发邮件至 zltts@phei.com.cn。

服务热线: (010) 88258888。



李刚

从事过6年的J2EE应用开发,曾担任LITEON公司的J2EE技术主管,负责该公司的企业信息平台的架构设计。担任过广东龙泉科技有限公司的J2EE技术培训导师。目前在新东方IT培训中心担任J2EE培训讲师,培训的学生已经在广州、深圳等多家软件公司就业,在珠三角的J2EE行业极具号召力。

有个老人在河边钓鱼，一个小孩走过去看他钓鱼。老人技巧纯熟，所以没多久就钓上了满篓的鱼。老人见小孩很可爱，要把整篓的鱼送给他，小孩摇摇头，老人惊异地问道：“你为何不要？”

小孩回答：“我想要你手中的钓竿。”

老人问：“你要钓竿做什么？”

小孩说：“这篓鱼没多久就吃完了，要是我有钓竿，我就可以自己钓，一辈子也吃不完。”

我想你一定会说：好聪明的小孩。

错了，他如果只要钓竿，那他一条鱼也吃不到。因为，他不懂钓鱼的技巧，光有钓竿是没用的，因为钓鱼重要的不在“钓竿”，而在“钓技”。有太多人认为自己拥有了人生道上的钓竿，再也无惧于路上的风雨，如此，难免会跌倒于泥泞地上。就如小孩看老人，以为只要有钓竿就有吃不完的鱼，像职员看老板，以为只要坐在办公室，就有滚进的财源。

反侵权盗版声明

电子工业出版社依法对本作品享有专有出版权。任何未经权利人书面许可，复制、销售或通过信息网络传播本作品的行为；歪曲、篡改、剽窃本作品的行为，均违反《中华人民共和国著作权法》，其行为人应承担相应的民事责任和行政责任，构成犯罪的，将被依法追究刑事责任。

为了维护市场秩序，保护权利人的合法权益，我社将依法查处和打击侵权盗版的单位和个人。欢迎社会各界人士积极举报侵权盗版行为，本社将奖励举报有功人员，并保证举报人的信息不被泄露。

举报电话：(010) 88254396; (010) 88258888

传 真：(010) 88254397

E-mail: dbqq@phei.com.cn

通信地址：北京市万寿路173信箱
电子工业出版社总编办公室

邮 编：100036

前 言

Spring 在 2003 年一经推出，即引起 J2EE 领域极大的兴趣，成为风头最健的 J2EE 开源框架之一。Spring 的版本更新非常快，目前的最新版本是 2.0。Spring 倡导“实用主义”原则，简化 J2EE 应用的开发。Spring 对 J2EE 各层组件都提出了对应的解决方案，它的 IoC 容器则对组件之间的依赖充分解耦，强化面向接口编程的概念。

为了帮助众多 Java 学习、工作者提高 J2EE 应用开发能力，笔者精心编著了本书。本书按 Spring 的架构体系，细致地介绍 Spring 各个知识点。在介绍过程中，笔者依照读者的学习规律，首先介绍基本概念和基本操作，然后对内容进行深入讲解。笔者不仅介绍 Spring 的相关知识点，更加注重灌输关于 J2EE 的架构知识。本书既是一本介绍 Spring 框架的书籍，也是一本关于 J2EE 架构设计的指南。

笔者作为一名资深的程序员，早年也曾经历技术学习的苦痛，层出不穷的新名词让人眼花缭乱，特别是软件行业知识更新快，常常让人无所适从。笔者在担任新东方 IT 培训中心 J2EE 讲师期间，与同学们的压抑感同身受，过多的理论，过多的新名词，过多的专业术语对初学者来说是一场灾难。

大量实际开发工作的经验，担任软件公司培训导师的经历，从事职业培训的过程给笔者一个概念——软件开发偏重于实践，而不是理论。笔者不想显摆自己的知识渊博，给读者罗列一串一串的新名词。对于各种概念，本书希望能以最浅显的语言，表达出最传神的内涵。本书不惜“化神奇为腐朽”，将各种专业的、深奥的概念变得更加直白。

本书不想以空洞的说教告诉读者——各个类的用处，各个方法的用法；而是以实际的示例，结合 Spring 的 API，演示 Spring 的用法。所有的示例都是笔者精心挑选，针对特定的知识点设计的。示例往往直观易懂，却充分演示了对应知识点的用法。目前市场上已经有了几本 Spring 的书籍，和这些书籍相比，本书有如下优点。

1. 经验丰富，针对性强

笔者既担任过软件开发的技术经理，也担任过软件公司的培训导师，也从事过职业培训的专职讲师。熟悉软件开发过程的难点，同时也理解软件学习过程中的苦楚。因此，本书尽量针对那些在学习过程中容易遇到的苦痛，重点讲解；对开发过程中的“陷阱”，则给出提示。

2. 示例丰富，示范性强

本书的各示例针对不同的知识点设计，尽量不与其他知识点掺杂在一起，混淆读者注意力。各知识点的示例重点突出，示范性强。

3. 知识全面，涉及面广

在介绍 Spring 框架的同时，还介绍与 Spring 关联的大量知识，如 Struts, WebWork2, JSF, Velocity, XSLT, POI, iText, JavaMail, Quartz 等。因此，本书不仅可作为 Spring 的参考手册使用，还可作为 J2EE 应用的开发指南。

4. 代码详细，注视全面

本书的代码注释十分到位，适合初学者学习，无论您从哪里开始阅读，都可以快速明白书中所表达的意思，学到想要的内容。

5. 配有综合案例，快速提高

本书最后配有两个典型的案例，让读者在掌握基础知识的情况下，迅速提升到项目开发级别，应

用于项目实践。

本书包括的内容:

- ◆ Spring 的核心机制, 包括 Spring 的 IoC 容器介绍, Spring 的 Bean 管理等。
- ◆ Spring 提供的资源访问机制, 包括访问本地资源, 网络资源和 web 应用中的资源。
- ◆ Spring 提供的 AOP 框架介绍, 包括 AOP 的基本知识, Spring 的 AOP 编程。
- ◆ Spring 提供的 JDBC 支持介绍, 包括对 JDBC 的简化。
- ◆ Spring 提供的 DAO 支持介绍, 包括各种 DAO 支持类。
- ◆ Spring 的 MVC 框架介绍, 包括 MVC 介绍等知识。
- ◆ Spring 与 Struts, WebWork, JSF 的整合。
- ◆ Spring 整合 Velocity, JSTL, XSLT, POI, iText 等。
- ◆ Spring 的远程访问支持, 包括 WebService, Hessian, RMI 等知识的介绍。
- ◆ Spring 的 EJB 访问支持。
- ◆ Spring 对 JMS 的简化, 包括消息驱动 EJB 的介绍。
- ◆ Spring 的邮件抽象层介绍, 包括 JavaMail 等知识。
- ◆ Spring 中的任务调度介绍, 包括 JDK 的 Timer 和 OpenSymphony 的 Quartz 的介绍。
- ◆ Spring 的测试框架介绍, 包括测试的基础知识、Spring 的 mock 对象, 以及利用 DI 测试业务组件等知识。

本书适用于有较好的 Java 编程基础, 有一定的 J2EE 编程基础的用户。本书的内容非常全面, 既可以作为 Spring 学习者的学习指南, 也可以作为实际开发者的参考手册。

李 刚
2006 年 9 月

目 录

| | |
|--|----|
| 第 1 部分 Spring 核心技术 | 1 |
| 第 1 章 Spring 概述 | 2 |
| 1.1 Spring 的起源和背景 | 2 |
| 1.2 Spring 初探 | 3 |
| 1.2.1 运行 Spring 所需的环境 | 3 |
| 1.2.2 Spring 的下载和安装 | 4 |
| 1.3 Spring 体系介绍 | 9 |
| 1.3.1 Spring 的核心和 Context | 9 |
| 1.3.2 Spring 的 Web 和 MVC | 9 |
| 1.3.3 Spring 的面向切面编程 | 10 |
| 1.3.4 Spring 的持久化支持 | 10 |
| 1.4 Spring 的基本设计思想 | 10 |
| 1.4.1 单态模式的回顾 | 10 |
| 1.4.2 工厂模式的回顾 | 11 |
| 1.4.3 Spring 对单态与工厂模式的实现 | 14 |
| 1.5 Spring 的核心机制 | 15 |
| 1.5.1 理解依赖注入 | 15 |
| 1.5.2 设值注入 | 16 |
| 1.5.3 构造注入 | 20 |
| 1.5.4 两种注入方式的对比 | 21 |
| 1.6 小结 | 21 |
| 第 2 章 Spring 中的 bean 和 BeanFactory | 22 |
| 2.1 bean 简介 | 22 |
| 2.2 bean 定义和 bean 实例化 | 22 |
| 2.2.1 BeanFactory 接口介绍 | 22 |
| 2.2.2 在 Context 定义 bean | 24 |
| 2.2.3 bean 的基本行为 | 25 |
| 2.2.4 bean 与 JavaBean 的关系 | 26 |
| 2.2.5 实例化 bean | 28 |
| 2.3 bean 特性的深入 | 35 |
| 2.3.1 bean 的高级属性、合作者 | 36 |
| 2.3.2 使用 depends-on 强制初始化 bean | 41 |
| 2.3.3 自动装配 | 41 |
| 2.3.4 依赖检查 | 44 |
| 2.4 bean 的生命周期 | 46 |
| 2.4.1 了解 bean 的生命周期 | 46 |
| 2.4.2 协调不同步的 bean | 46 |
| 2.4.3 定制 bean 的生命周期行为 | 49 |
| 2.5 小结 | 52 |
| 第 3 章 bean 的高级功能 | 53 |
| 3.1 bean 的继承 | 53 |
| 3.1.1 抽象 bean | 53 |
| 3.1.2 定义子 bean | 54 |

- 3.1.3 Spring 中 bean 的继承与 Java 中继承的区别 56
- 3.2 高级依赖注入 56
 - 3.2.1 属性值的依赖注入 56
 - 3.2.2 field 值的依赖注入 59
 - 3.2.3 方法返回值的依赖注入 60
- 3.3 使用 BeanPostProcessor 63
- 3.4 使用 BeanFactoryPostProcessor 65
 - 3.4.1 使用属性占位符配置器 66
 - 3.4.2 另一种属性占位符配置器 PropertyOverrideConfigurer 67
- 3.5 与容器交互 68
 - 3.5.1 工厂 bean 简介与配置 68
 - 3.5.2 使用 FactoryBean 接口 70
 - 3.5.3 使用 BeanFactoryAware 获取 BeanFactory 72
 - 3.5.4 使用 BeanNameAware 回调本身 74
- 3.6 ApplicationContext 介绍 75
 - 3.6.1 国际化支持 75
 - 3.6.2 事件处理 77
 - 3.6.3 Web 应用中自动加载 ApplicationContext 79
- 3.7 汇总多个 XML 配置文件 80
 - 3.7.1 ApplicationContext 加载多个配置文件 80
 - 3.7.2 Web 应用启动时加载多个配置文件 81
 - 3.7.3 XML 配置文件中导入其他配置文件 81
- 3.8 小结 81
- 第 4 章 Spring 中的资源访问 82**
 - 4.1 传统资源访问与 Spring 的资源访问 82
 - 4.1.1 传统资源访问 82
 - 4.1.2 Spring 的资源访问 83
 - 4.2 Resource 实现类 85
 - 4.2.1 访问网络资源 85
 - 4.2.2 使用 ClassPathResource 86
 - 4.2.3 访问文件系统资源 87
 - 4.2.4 访问应用相关资源 88
 - 4.2.5 访问输入流资源 90
 - 4.2.6 访问字节数组资源 92
 - 4.3 ResourceLoader 接口和 ResourceLoaderAware 接口 93
 - 4.3.1 使用 ResourceLoader 93
 - 4.3.2 使用 ResourceLoaderAware 95
 - 4.4 使用 Resource 作为属性 96
 - 4.5 ApplicationContext 中使用资源 99
 - 4.5.1 实例化 ApplicationContext 99
 - 4.5.2 classpath* 前缀的用法 101
 - 4.5.3 访问文件系统 103
 - 4.6 小结 104
- 第 5 章 表现层数据的处理 105**
 - 5.1 表现层数据涉及的处理 105
 - 5.1.1 类型转换 105
 - 5.1.2 数据校验 108
 - 5.2 Spring 支持的表现层数据处理 112
 - 5.2.1 数据绑定 112
 - 5.2.2 bean 包装 113

| | | |
|--------------|--|------------|
| 5.2.3 | 数据校验 | 113 |
| 5.3 | Bean 包装详解 | 113 |
| 5.3.1 | 修改、获取 Bean 属性 | 113 |
| 5.3.2 | 类型转换 | 116 |
| 5.3.3 | 内建的 PropertyEditor | 118 |
| 5.3.4 | 自定义 PropertyEditor | 119 |
| 5.4 | 数据校验详解 | 121 |
| 5.4.1 | 实现校验器 | 122 |
| 5.4.2 | 显示出错提示 | 123 |
| 5.5 | 小结 | 125 |
| 第 6 章 | Spring 对 AOP 的支持 | 126 |
| 6.1 | AOP 入门 | 126 |
| 6.1.1 | AOP 中的概念 | 126 |
| 6.1.2 | AOP 代理 | 127 |
| 6.2 | Spring 对 AOP 的支持 | 130 |
| 6.2.1 | 简介 | 130 |
| 6.2.2 | Spring 的切入点 | 131 |
| 6.2.3 | Spring 的处理 | 134 |
| 6.3 | 创建 AOP 代理 | 150 |
| 6.3.1 | 基本概念 | 150 |
| 6.3.2 | 代理接口 | 151 |
| 6.3.3 | 代理类 | 155 |
| 6.4 | 实用的代理工厂类 | 155 |
| 6.4.1 | 使用 TransactionProxyFactoryBean | 156 |
| 6.4.2 | 使用 LocalStatelessSessionProxyFactoryBean | 158 |
| 6.5 | 简洁的代理定义方式 | 160 |
| 6.6 | 自动代理 | 161 |
| 6.6.1 | 定义自动代理 bean | 161 |
| 6.6.2 | 自动代理的实现 | 161 |
| 6.7 | 编程式创建 AOP 代理 | 164 |
| 6.8 | 操作代理 | 166 |
| 6.9 | 小结 | 167 |
| 第 7 章 | Spring 的事务管理 | 168 |
| 7.1 | 事务介绍 | 168 |
| 7.1.1 | 事务的基本概念 | 168 |
| 7.1.2 | 事务的特性 | 168 |
| 7.2 | Spring 中的事务 | 169 |
| 7.2.1 | 传统事务的特征和弱点 | 169 |
| 7.2.2 | Spring 事务的优势 | 169 |
| 7.3 | 使用事务管理器接口 PlatformTransactionManager | 170 |
| 7.4 | 编程式事务 | 174 |
| 7.4.1 | 使用 TransactionTemplate | 174 |
| 7.4.2 | 使用 PlatformTransactionManager | 176 |
| 7.5 | 声明式事务 | 177 |
| 7.5.1 | 使用声明式事务管理 | 177 |
| 7.5.2 | 根据 BeanName 自动创建事务代理 | 180 |
| 7.5.3 | 基于 JDK1.5+ 的注释式事务代理配置 | 182 |
| 7.5.4 | 使用 bean 继承简化事务代理配置 | 185 |
| 7.6 | 关于事务管理的思考 | 186 |
| 7.6.1 | 编程式事务 VS 声明式事务 | 186 |

| | | |
|--------|--|-----|
| 7.6.2 | 应用服务器管理事务 | 187 |
| 7.7 | 小结 | 187 |
| 第 2 部分 | Spring 与其他工具、框架整合应用 | 189 |
| 第 8 章 | DAO 支持 | 190 |
| 8.1 | DAO 模式介绍 | 190 |
| 8.1.1 | J2EE 应用的通用分层 | 190 |
| 8.1.2 | 编写 DAO | 192 |
| 8.1.3 | 如何使用 DAO | 196 |
| 8.2 | Spring 中 DAO 的体系 | 197 |
| 8.2.1 | 统一的异常继承体系 | 197 |
| 8.2.2 | 统一的 DAO 抽象 | 198 |
| 8.3 | 常见的 DAO 支持类 | 198 |
| 8.3.1 | Spring 对 JDBC 的 DAO 支持 | 199 |
| 8.3.2 | Spring 对 Hibernate 的 DAO 支持 | 202 |
| 8.4 | DAO 模式的异常处理 | 206 |
| 8.4.1 | 编写 DAO 异常类 | 206 |
| 8.4.2 | 改写 DAO 实现类 | 207 |
| 8.5 | 小结 | 209 |
| 第 9 章 | 使用 JDBC 操作数据库 | 210 |
| 9.1 | JDBC 基础 | 210 |
| 9.1.1 | JDBC 简介 | 210 |
| 9.1.2 | JDBC 驱动 | 210 |
| 9.1.3 | JDBC 常用接口和类简介 | 211 |
| 9.1.4 | 传统的 JDBC 访问数据库 | 213 |
| 9.2 | 数据库连接池 | 213 |
| 9.2.1 | 数据库连接池介绍 | 213 |
| 9.2.2 | 常见的数据库连接池 | 217 |
| 9.3 | Spring 的 JDBC 体系 | 221 |
| 9.3.1 | Spring 的 JDBC 封装 | 221 |
| 9.3.2 | Spring 的 JDBC 与传统 JDBC 的对比 | 222 |
| 9.4 | 使用 JdbcTemplate 访问数据库 | 223 |
| 9.4.1 | 执行简单查询 | 223 |
| 9.4.2 | 执行简单更新 | 224 |
| 9.4.3 | 执行简单 DDL | 225 |
| 9.4.4 | 使用 StatementCallback 访问数据库 | 226 |
| 9.4.5 | 使用 PreparedStatementCallback 访问数据库 | 227 |
| 9.5 | 连接数据库的辅助类 | 228 |
| 9.5.1 | 使用数据源工具类 (DataSourceUtils) | 228 |
| 9.5.2 | 智能数据源 (SmartDataSource) 接口介绍 | 229 |
| 9.5.3 | 单连接数据源 SingleConnectionDataSource 的使用 | 229 |
| 9.5.4 | 另一个数据源实现: DriverManagerDataSource | 230 |
| 9.5.5 | 数据源的事务管理器 DataSourceTransactionManager | 231 |
| 9.6 | 数据库操作的对象化 | 233 |
| 9.6.1 | 查询结果对象化 | 233 |
| 9.6.2 | 查询对象 | 235 |
| 9.6.3 | 更新对象 | 237 |
| 9.6.4 | 使用 StoredProcedure 调用存储过程或函数 | 238 |
| 9.6.5 | 使用 StoredProcedure 调用函数 | 240 |
| 9.6.6 | 使用 SqlFunction 执行查询或调用函数 | 241 |

| | | |
|---------------|--|------------|
| 9.7 | 小结 | 243 |
| 第 10 章 | 整合 Hibernate 执行持久化操作 | 244 |
| 10.1 | ORM 介绍 | 244 |
| 10.1.1 | 什么是 ORM | 244 |
| 10.1.2 | 为什么需要 ORM | 244 |
| 10.1.3 | 流行的 ORM 框架简介 | 245 |
| 10.2 | Hibernate 介绍 | 245 |
| 10.2.1 | 采用传统 JDBC 操作数据库 | 245 |
| 10.2.2 | Hibernate 下载和安装 | 247 |
| 10.2.3 | Hibernate 初探 | 247 |
| 10.2.4 | Hibernate 的基本映射 | 250 |
| 10.2.5 | Hibernate 的关系映射 | 251 |
| 10.2.6 | Hibernate 的 HQL 查询 | 257 |
| 10.2.7 | Hibernate 的 Criteria 查询 | 261 |
| 10.3 | 整合 Hibernate | 261 |
| 10.4 | 管理 SessionFactory | 262 |
| 10.5 | Spring 对 Hibernate 的简化 | 263 |
| 10.6 | 使用 HibernateTemplate | 264 |
| 10.6.1 | HibernateTemplate 的常规用法 | 266 |
| 10.6.2 | Hibernate 的复杂用法 HibernateCallback | 267 |
| 10.7 | Hibernate 的 DAO 实现 | 268 |
| 10.7.1 | 继承 HibernateDaoSupport 实现 DAO | 268 |
| 10.7.2 | 基于 Hibernate 3.0 实现 DAO | 271 |
| 10.8 | 事务管理 | 272 |
| 10.8.1 | 编程式的事务管理 | 272 |
| 10.8.2 | 声明式的事务管理 | 274 |
| 10.8.3 | 事务策略的思考 | 276 |
| 10.9 | 小结 | 276 |
| 第 11 章 | Spring 的 MVC 框架 | 277 |
| 11.1 | MVC 入门 | 277 |
| 11.1.1 | 传统的 Model 1 和 Model 2 | 277 |
| 11.1.2 | MVC 及其优势 | 278 |
| 11.2 | Web MVC 框架简介 | 279 |
| 11.2.1 | 目前流行的 MVC 框架 | 279 |
| 11.2.2 | MVC 框架的基本特征 | 280 |
| 11.3 | Spring MVC 的特点和优、劣势 | 280 |
| 11.4 | Spring MVC 框架中的角色 | 281 |
| 11.4.1 | 核心控制器 DispatcherServlet | 281 |
| 11.4.2 | 业务控制器 | 282 |
| 11.4.3 | 处理器映射 | 282 |
| 11.4.4 | 视图和视图解析器 | 282 |
| 11.4.5 | 模型 | 282 |
| 11.4.6 | Command 对象 | 282 |
| 11.5 | 核心控制器 | 282 |
| 11.6 | 控制器映射 | 283 |
| 11.6.1 | 使用 BeanNameUrlHandlerMapping | 283 |
| 11.6.2 | SimpleUrlHandlerMapping | 286 |
| 11.7 | 业务控制器 | 287 |
| 11.7.1 | AbstractController 和 WebContentGenerator | 287 |
| 11.7.2 | 简单控制器 | 288 |

| | | |
|---------------|--|------------|
| 11.7.3 | 使用 ParameterizableViewController | 288 |
| 11.7.4 | 使用 UriFilenameViewController | 289 |
| 11.7.5 | CommandController 类介绍 | 290 |
| 11.7.6 | 使用 SimpleFormController | 291 |
| 11.7.7 | 使用 MultiActionController | 293 |
| 11.8 | 模型 | 296 |
| 11.9 | 视图和视图解析器 | 296 |
| 11.9.1 | InternalResourceViewResolver 示例 | 297 |
| 11.9.2 | 视图解析器的链式处理 | 298 |
| 11.9.3 | 重定向视图 | 298 |
| 11.10 | Spring MVC 中的程序国际化 | 300 |
| 11.10.1 | 自动化的国际化信息解析 | 300 |
| 11.10.2 | 自定义的国际化信息解析 | 301 |
| 11.11 | 数据校验 | 302 |
| 11.12 | 文件上传 | 304 |
| 11.12.1 | 使用 MultipartResolver | 304 |
| 11.12.2 | 定义 Command 对象和控制器 | 305 |
| 11.13 | 小结 | 307 |
| 第 12 章 | 整合第三方 MVC 框架 | 308 |
| 12.1 | 目前流行的 MVC 框架分析 | 308 |
| 12.1.1 | Struts | 308 |
| 12.1.2 | WebWork2 | 309 |
| 12.1.3 | JSF | 309 |
| 12.1.4 | Tapestry | 309 |
| 12.2 | Spring 整合第三方 MVC 框架的通用配置 | 310 |
| 12.2.1 | 采用 ServletContextListener 创建 ApplicationContext | 310 |
| 12.2.2 | 采用 load-on-startup Servlet 创建 ApplicationContext | 312 |
| 12.3 | 整合 Struts | 313 |
| 12.3.1 | Struts 的使用 | 313 |
| 12.3.2 | Spring 管理 Struts 的 Action | 318 |
| 12.3.3 | 使用 ActionSupport 代替 Action | 325 |
| 12.4 | 整合 WebWork2 | 326 |
| 12.4.1 | 使用 Webwork2 | 327 |
| 12.4.2 | WebWork2 与 Spring 的整合 | 334 |
| 12.5 | 整合 JSF | 335 |
| 12.5.1 | 使用 JSF | 336 |
| 12.5.2 | JSF 的依赖注入 | 341 |
| 12.5.3 | 利用 JSF 的依赖注入整合 Spring | 342 |
| 12.5.4 | 利用 FacesContextUtils | 345 |
| 12.6 | 小结 | 345 |
| 第 13 章 | 整合第三方表现层技术 | 346 |
| 13.1 | 表现层介绍 | 346 |
| 13.1.1 | 表现层的概念 | 346 |
| 13.1.2 | 表现层的功能 | 346 |
| 13.1.3 | 常见的表现层技术 | 346 |
| 13.2 | 整合 JSTL | 347 |
| 13.2.1 | JSTL 介绍 | 347 |
| 13.2.2 | 使用 JSTLView 视图解析器 | 348 |
| 13.3 | 整合 Velocity | 350 |
| 13.3.1 | Velocity 介绍 | 351 |

| | | |
|---------------|--|------------|
| 13.3.2 | Spring 对 Velocity 的支持 | 353 |
| 13.3.3 | 管理 Velocity 的模板 | 355 |
| 13.3.4 | 处理表单 | 357 |
| 13.4 | 整合 XSTL 视图 | 358 |
| 13.4.1 | XML 知识简介 | 358 |
| 13.4.2 | XSTL 介绍 | 360 |
| 13.4.3 | Spring 对 XSLT 视图的支持 | 361 |
| 13.4.4 | 转换模型的数据 | 362 |
| 13.5 | 生成 Excel 表格或 PDF 文档 | 364 |
| 13.5.1 | POI 介绍 | 364 |
| 13.5.2 | iText 简介 | 367 |
| 13.5.3 | 生成 Excel 表格 | 369 |
| 13.5.4 | 创建 PDF 文档 | 371 |
| 13.6 | 小结 | 374 |
| 第 14 章 | 通过 Spring 进行远程访问和 WebService | 375 |
| 14.1 | 远程访问简介 | 375 |
| 14.1.1 | 远程访问的意义 | 375 |
| 14.1.2 | 常用的远程访问技术 | 376 |
| 14.2 | WebService 简介 | 376 |
| 14.2.1 | WebService 的特点 | 376 |
| 14.2.2 | WebService 的主要技术 | 378 |
| 14.3 | Spring 对远程访问的支持 | 379 |
| 14.4 | 通过 RMI 提供服务 | 379 |
| 14.4.1 | RMI 介绍 | 379 |
| 14.4.2 | 使用 RmiServiceExporter 提供服务 | 382 |
| 14.4.3 | 在客户端连接服务 | 384 |
| 14.5 | 使用 Hessian 通过 Http 提供服务 | 385 |
| 14.5.1 | Hessian 介绍 | 386 |
| 14.5.2 | 为 Hessian 装配 DispatcherServlet | 388 |
| 14.5.3 | 使用 HessianServiceExporter 提供 bean 服务 | 389 |
| 14.5.4 | 在客户端连接服务 | 390 |
| 14.6 | 通过 Web Service 提供服务 | 391 |
| 14.6.1 | WebService 的开源实现 | 391 |
| 14.6.2 | Spring 对 WebService 的支持 | 394 |
| 14.6.3 | 从客户端访问 Spring 发布的 WebService | 397 |
| 14.7 | 使用 httpInvoker 提供远程服务 | 398 |
| 14.7.1 | 输出业务对象 | 398 |
| 14.7.2 | 客户端连接服务 | 400 |
| 14.8 | 小结 | 401 |
| 第 15 章 | EJB 的访问和实现 | 402 |
| 15.1 | EJB 简介 | 402 |
| 15.1.1 | EJB 的概念和分类 | 402 |
| 15.1.2 | 如何编写 EJB | 403 |
| 15.1.3 | EJB 的部署与运行 | 406 |
| 15.1.4 | 程序中调用 EJB | 406 |
| 15.2 | 传统 EJB 访问的讨论 | 408 |
| 15.2.1 | 常规的 EJB 访问策略 | 408 |
| 15.2.2 | 传统 EJB 访问的不足 | 410 |
| 15.2.3 | Spring 的 EJB 访问策略 | 411 |
| 15.3 | 利用 Spring 简化 EJB 的访问 | 411 |

| | | |
|---------------|---|------------|
| 15.3.1 | 访问 Local 无状态会话 Bean | 411 |
| 15.3.2 | 访问 Remote 无状态会话 Bean | 415 |
| 15.4 | Spring 提倡的 EJB 架构 | 420 |
| 15.4.1 | 使用 DI 将 EJB 的业务逻辑委托给 POJO | 420 |
| 15.4.2 | 管理 EJB 中的 Spring 容器 | 422 |
| 15.5 | 小结 | 423 |
| 第 16 章 | Spring 中使用 JMS | 424 |
| 16.1 | JMS 介绍 | 424 |
| 16.1.1 | JMS 简介 | 424 |
| 16.1.2 | JMS 开发 | 425 |
| 16.2 | 消息驱动 EJB 介绍 | 435 |
| 16.2.1 | 消息驱动 EJB 的 bean 类 | 435 |
| 16.2.2 | 消息驱动 EJB 的配置文件 | 437 |
| 16.3 | Spring 对 JMS 的支持 | 438 |
| 16.3.1 | Spring 的 JmsTemplate | 439 |
| 16.3.2 | 管理连接工厂 | 439 |
| 16.3.3 | 管理消息目的 | 440 |
| 16.3.4 | JMS 与事务 | 440 |
| 16.4 | 通过 Spring 发送消息 | 440 |
| 16.4.1 | 使用 JmsTemplate 的发送消息 | 440 |
| 16.4.2 | 发送消息的配置文件 | 441 |
| 16.4.3 | 完成消息的发送 | 443 |
| 16.5 | 通过 Spring 接收消息 | 443 |
| 16.5.1 | 使用 JmsTemplate 的接收消息 | 443 |
| 16.5.2 | 接收消息的配置文件 | 444 |
| 16.5.3 | 完成消息的接收 | 445 |
| 16.6 | Pub/Sub 模型的 JMS 消息处理 | 445 |
| 16.6.1 | 修改后的消息发送文件 | 446 |
| 16.6.2 | 消息发送的配置 | 447 |
| 16.6.3 | 修改后的消息接收文件 | 447 |
| 16.6.4 | 消息接收的配置 | 448 |
| 16.7 | 小结 | 449 |
| 第 17 章 | 利用 Spring 发送邮件 | 450 |
| 17.1 | E-mail 简介 | 450 |
| 17.1.1 | SMTP 协议简介 | 450 |
| 17.1.2 | POP3 协议简介 | 451 |
| 17.1.3 | E-mail 的用处 | 451 |
| 17.2 | JavaMail 介绍 | 451 |
| 17.2.1 | JavaMail 下载和安装 | 452 |
| 17.2.2 | JavaMail 的使用 | 452 |
| 17.3 | Spring 的邮件抽象体系 | 458 |
| 17.4 | 使用 Spring 的邮件体系发送邮件 | 459 |
| 17.4.1 | 使用 MailSender 发送简单邮件 | 459 |
| 17.4.2 | 使用 JavaMailSender 发送 MimeMessage 信息 | 460 |
| 17.4.3 | 综合应用 | 463 |
| 17.5 | 小结 | 468 |
| 第 18 章 | Spring 中的任务调度 | 469 |
| 18.1 | 任务调度的概念 | 469 |

| | | |
|---------------|--|------------|
| 18.1.1 | 任务调度简介 | 469 |
| 18.1.2 | 任务调度的作用 | 469 |
| 18.1.3 | 常见的任务调度支持类 | 469 |
| 18.2 | JDK Timer 介绍 | 470 |
| 18.2.1 | 建立任务 | 470 |
| 18.2.2 | 调度任务 | 471 |
| 18.3 | OpenSymphony 的 Quartz 介绍 | 472 |
| 18.3.1 | Quartz 的下载和安装 | 472 |
| 18.3.2 | Quartz 的使用 | 473 |
| 18.4 | 在 Spring 中使用 Timer | 477 |
| 18.4.1 | 继承 TimerTask 创建任务 | 477 |
| 18.4.2 | 使用 MethodInvokingTimerTaskFactoryBean 创建任务 | 480 |
| 18.5 | 在 Spring 中使用 Quartz | 484 |
| 18.5.1 | 继承 QuartzJobBean 创建作业 | 484 |
| 18.5.2 | 使用 MethodInvokingJobDetailFactoryBean 工厂 bean 创建作业 | 488 |
| 18.6 | 小结 | 490 |
| 第 19 章 | 利用 Spring 简化测试 | 491 |
| 19.1 | 软件测试介绍 | 491 |
| 19.1.1 | 什么是软件测试 | 491 |
| 19.1.2 | 软件测试的目的 | 492 |
| 19.1.3 | 测试分类 | 492 |
| 19.2 | JUnit 介绍 | 493 |
| 19.2.1 | 单元测试概述 | 493 |
| 19.2.2 | JUnit 概述 | 493 |
| 19.2.3 | JUnit 的下载和安装 | 494 |
| 19.2.4 | JUnit 中常用的接口和类 | 494 |
| 19.2.5 | JUnit 的使用 | 495 |
| 19.3 | 利用 Spring 的 mock 进行单元测试 | 499 |
| 19.3.1 | Spring 的 mock 类 | 500 |
| 19.3.2 | 利用 mock 类测试控制器 | 500 |
| 19.4 | 利用 DI 完成集成测试 | 503 |
| 19.4.1 | Spring 的辅助测试类 | 503 |
| 19.4.2 | 持久层组件的测试 | 503 |
| 19.4.3 | 业务层组件测试 | 508 |
| 19.5 | 小结 | 512 |
| 第 3 部分 | Spring 典型案例 | 513 |
| 第 20 章 | 完整实例：新闻发布系统 | 514 |
| 20.1 | 系统架构说明 | 514 |
| 20.1.1 | 架构处理流程框架图 | 514 |
| 20.1.2 | 系统架构说明 | 515 |
| 20.1.3 | 简单的处理流程示例 | 515 |
| 20.1.4 | 对系统架构的疑问 | 516 |
| 20.2 | Domain 层 | 516 |
| 20.2.1 | 编写 Domain Object | 516 |
| 20.2.2 | 编写 PO 的映射配置文件 | 522 |
| 20.2.3 | Spring 整合 Hibernate | 524 |
| 20.3 | 持久层 | 525 |
| 20.3.1 | DAO 模式的简单介绍 | 525 |
| 20.3.2 | 使用 DAO 模式的原因 | 526 |