

◆附赠全部源码◆

# Java EE

## 互联网轻量级框架整合开发

SSM框架（Spring MVC+Spring+MyBatis）和Redis实现

杨开振 周吉文 梁华辉 谭茂华 / 著

详述MyBatis和Spring MVC，深入Spring数据库事务机制  
通过Spring介绍Redis，以抢红包介绍高并发与锁



中国工信出版集团



电子工业出版社  
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY  
http://www.phei.com.cn



# Java EE

## 互联网轻量级框架整合开发

SSM框架 ( Spring MVC+Spring+MyBatis ) 和Redis实现

杨开振 周吉文 梁华辉 谭茂华 / 著

电子工业出版社

Publishing House of Electronics Industry

北京·BEIJING

## 内 容 简 介

随着移动互联网的兴起，以 Java 技术为后台的互联网技术占据了市场的主导地位，而在 Java 互联网后台开发中，SSM 框架（Spring+Spring MVC+MyBatis）成为了主要架构，本书以此为焦点从入门到实际工作要求讲述了 SSM 框架的技术应用；与此同时，为了提高系统性能，NoSQL（尤其是 Redis）在互联网系统中已经广泛使用，为了适应这个变化，本书通过 Spring 讲解了有关 Redis 的技术应用，这样更加贴近实际学习和工作的需要。

本书主要分为 6 个部分，第 1 部分对 Java 互联网的框架和主要涉及的模式做初步简介；第 2 部分讲述 MyBatis 技术；第 3 部分讲述 Spring 基础（包括 IoC、AOP 和数据库应用），重点讲解 Spring 数据库事务应用，以满足互联网企业的应用要求；第 4 部分，讲述 Spring MVC 框架；第 5 部分，通过 Spring 技术的应用，讲解 Redis 技术；第 6 部分，讲解 SSM+Redis 实践应用，通过互联网高并发如抢票、抢红包等场景，使用全注解的方式讲解 SSM 框架的整合，以及高并发与锁的应用和系统性能优化。

本书结合企业的实际需求，从原理到实践全面讲解 SSM+Redis 技术应用，无论你是 Java 程序员、SSM 应用和研究人员，还是 Redis 应用人员、互联网开发人员，都可以从本书中收获知识。

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

Java EE 互联网轻量级框架整合开发：SSM 框架（Spring MVC+Spring+MyBatis）和 Redis 实现 / 杨开振等著. —北京：电子工业出版社，2017.7

ISBN 978-7-121-31847-4

I. ①J… II. ①杨… III. ①JAVA 语言—程序设计 ②数据库—基本知识 IV. ①TP312.8 ②TP311.138

中国版本图书馆 CIP 数据核字（2017）第 130168 号

策划编辑：汪达文

责任编辑：徐津平

印 刷：三河市鑫金马印装有限公司

装 订：三河市鑫金马印装有限公司

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：43 字数：1100 千字

版 次：2017 年 7 月第 1 版

印 次：2017 年 8 月第 2 次印刷

印 数：3001~4000 册 定价：119.00 元

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，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：（010）51260888-819，faq@phei.com.cn。

## 作者简介

- 杨开振

拥有十余年一线Java开发工作经验，热衷于Java互联网技术方向研究，目前从事Java互联网技术在线教育，著有畅销书《深入浅出MyBatis技术原理与实战》。

- 周吉文

高级工程师，曾获广东省科技进步一等奖，曾任国家“863”计划项目软件组组长，担任珠海市第二人民医院信息部主任多年，有丰富的项目实施和管理经验。

- 梁华辉

资深Java开发专家，丰富的企业级应用及Web项目的开发经验。对软件设计模式和代码整洁方面有深入理解，对Spring、Hibernate等主流开源框架有深入研究。

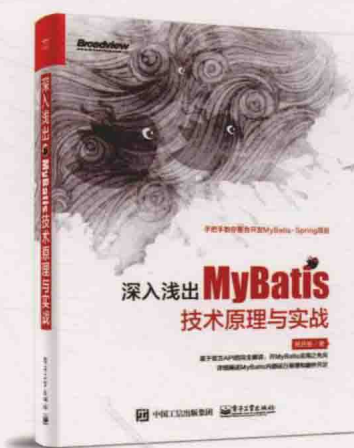
- 谭茂华

资深Java开发专家，拥有十余年供应链管理、金融、物联网等行业大型系统软件开发经验。热衷于软件架构、软件研发过程控制、Java开源技术（尤其是Spring体系技术）的研究。



扫一扫，加入本书共读群

## 好书分享



《深入浅出MyBatis技术原理与实战》  
ISBN 978-7-121-29594-2  
定价：69.00元



欢迎反馈意见或投稿

邮箱：wangdw@phei.com.cn

电话：010-88254356-858

qq号：381712977

# 前言

随着移动互联网的兴起以及手机和平板电脑的普及，Java 开发方向发生了很大变化，渐渐从管理系统走向了互联网系统。互联网系统的要求是大数据、高并发、高响应，而非管理系统的少数据、低并发和缓慢响应。为顺应技术发展趋势，2016 年春季笔者写了一本关于 MyBatis 的著作《深入浅出 MyBatis 技术原理与实战》，作为国内第一本关于 MyBatis 技术的著作，该书受到了业内的广泛肯定。与此同时，电子工业出版社编辑汪达文给了我一个很好的建议，她建议写一本 Java 合集，这更贴近开发工作的实际需求。在移动互联网兴起的时代，Spring+Spring MVC+MyBatis (SSM) 的 Java 组合已经成为时代的主流，伴随着 NoSQL (缓存) 的广泛应用，Redis 成了主要的 NoSQL 工具，这些都是当今移动互联网最为流行的技术，于是笔者打算竭尽所能写一本 SSM+Redis 的合集，这就是本书创作的缘起。

## 移动互联网的新要求

- 高并发：举个例子，大公司企业 ERP 应用，有 1 万名员工使用，同时在线的用户可能只有数百人，而操作一个业务的同一个数据的可能只有几个人，其系统一般不存在高并发的压力，使用传统程序和数据库完全可以应付。在互联网中一件热门的商品，比如新版小米手机，可能刚一上市就有成千上万的请求到达服务器，要求瞬间执行数以万计的数据操作，对性能要求高，操作不当容易造成网站瘫痪，引发网站的生存危机。
- 高响应：企业管理系统可以缓慢处理一些业务，而在高并发的互联网系统中，却不可以，按照互联网的要求一般以 5 秒为上限，超过 5 秒后响应，用户体验不好，从而影响用户忠诚度，而这些往往需要在高并发和大数据量的场景下实现。
- 数据一致性：由于高并发，多个线程对同一数据同时访问，需要保证数据的一致性，比如电商网站的金额、商品库存不能出错，还要保证其性能不能太差，这是在管理系统中不会出现的场景。

- 技术复杂化: 在互联网中流行许多新技术, 比如常见的 NoSQL (Redis、MongoDB), 又如 MQ、RPC 框架、ZooKeeper、大数据、分布式等技术。

## 为什么选择 SSM 框架+Redis 的开发模式

首先, Struts2 框架和 Spring 结合, 多年来改变不了臃肿的老毛病, 更为严重的是近年来多次出现的漏洞问题, 使得其名声和使用率大降。这个时候 Spring MVC 框架成了新一代 MVC 框架的主流。它原生于 Spring 框架, 可以无缝对接 Spring 的核心技术。与 Struts 不同, 它的流程模块化, 没有那么多臃肿的类, 所以互联网应用的框架大部分使用的是 Spring MVC。

其次, 目前企业的 Java 应用中, Spring 框架是必须的, Spring 的核心是 IoC (控制反转), 它是一个大容器, 方便组装和管理各类系统内外部资源, 同时支持 AOP (面向切面编程), 这是对面向对象的补充, 目前广泛用于日志和数据库事务控制, 减少了大量的重复代码, 使得程序更为清晰。因为 Spring 可以使模块解耦, 控制对象之间的协作, 所以 Spring 框架是目前 Java 最为流行的框架, 几乎没有之一。

最后, 对于 Hibernate 而言, 笔者感慨最多, 在需要存储过程或者复杂 SQL 时, 它的映射关系几乎完全用不上, 所有的问题都需要自己敲代码处理。作为全映射的框架, 它的致命缺点是没有办法完全掌控数据库的 SQL, 而优化 SQL 是高并发、高响应系统的必然要求, 这是互联网系统的普遍特性, 所以 Hibernate 在互联网系统中被排除了。而另一个持久层框架 MyBatis, 它需要编写 SQL、提供映射规则, 不过它加入了动态 SQL、自动映射、接口编程等功能使得它简单易用, 同时支持 SQL 优化、动态绑定, 并满足高并发和高响应的要求, 所以它成为最流行的 Java 互联网持久框架。

NoSQL 的成功在于, 首先它是基于内存的, 也就是数据放在内存中, 而不是像数据库那样把数据放在磁盘上, 而内存的读取速度是磁盘读取速度的几十倍到上百倍, 所以 NoSQL 工具的速度远比数据库读取速度要快得多, 满足了高响应的要求。即使 NoSQL 将数据放在磁盘中, 它也是一种半结构化的数据格式, 读取到解析的复杂度远比数据库要简单, 这是因为数据库存储的是经过结构化、多范式等有复杂规则的数据, 还原为内存结构的速度较慢。NoSQL 在很大程度上满足了高并发、快速读/写和响应的要求, 所以它也是 Java 互联网系统的利器。于是两种 NoSQL 的工具 Redis 和 MongoDB 流行起来, 尤其是 Redis 已经成为了主要的 NoSQL 工具, 本书会详细介绍它的常用方法。

基于以上原因, Spring+Spring MVC +MyBatis 已经成了 Java 互联网时代的主流框架, 而 Redis 缓存已经成了主流的 NoSQL 技术, 笔者愿意将自己所掌握的知识分享给大家, 为目前奋斗在 SSM 和 Redis 战线上的同行们奉献一本有价值的参考书, 给一些准备进入这个行业的新手一定的帮助和指导。

## 本书的特点

全书具备五大特点。

- 实用性：全书内容来自于笔者多年互联网实践开发工作，理论结合实际应用。
- 理论性：突出基础理念，结合设计模式阐述框架的实现原理和应用理念，让读者知其然也知其所以然。
- 与时俱进：介绍最新框架技术，与当前互联网企业保持同步，比如全注解搭建 SSM 框架和 Redis 的应用，使得读者能够把最新技术应用到实际的工作中去。
- 突出热点和重点：着重介绍 MyBatis 实践应用，Spring 数据库及事务应用，使用 Spring 介绍 Redis 实践应用、高并发和锁等互联网热门技术的热点和重点。
- 性能要求突出：这是移动互联网的要求，因为互联网面对大数据和高并发，体现互联网企业真实需要。

## 本书的内容安排

本书基于一线企业的实际应用要求，介绍了 Java 互联网最流行的框架技术，内容全面，以实际应用为导向，取舍明确，尤其对于技术的重点、难点解释得深入浅出，案例丰富，具体来说本书在体例上分为六大部分。

第 1 部分，首先讲解基础，让读者对 SSM 框架里的每一门技术的主要作用有所了解。然后介绍 SSM 框架的主要设计模式，它们有助于从底层深入理解框架。

第 2 部分，讲解 MyBatis 的基础应用，包括其主要组成、配置、映射器、动态 SQL，并且深入 MyBatis 的底层运行原理和插件，详细讨论它们的高级应用。

第 3 部分，讲解 Spring IoC 和 Spring AOP。掌握 Spring 如何通过 IoC 管理资源，然后通过设计模式讨论 AOP 的实现原理及其使用方法、实践。讨论 Spring 对数据库的支持，如何整合 MyBatis，并且着重讨论了 Spring 数据库事务的相关内容，包括数据库隔离级别和传播行为的应用。

第 4 部分，讲解 Spring MVC 主要的流程、HandlerMapping 的应用、控制器 Controller、处理适配器（HandlerAdapter）、视图和视图解析器，然后讨论传递参数、注解、数据校验、消息转换和国际化等应用。

第 5 部分，掌握 NoSQL 的优势和应用方法，掌握 Redis 的常用数据类型和主要命令，以及一些基本的特性（比如事务）和用法，并教会你在 Java 和 Spring 环境中使用它。

第 6 部分，SSM 框架+Redis 的实战，通过全注解的方式搭建 SSM 框架，讲解 Redis 应用，并展现了互联网的核心问题——高并发和锁的问题。介绍了通过悲观锁、乐观锁和 Redis Lua 语言方案来解决高并发和锁的问题。



## 和读者的约定

为了方便论述，我们进行以下约定。

- import 语句一般不出现在代码中，主要是为了缩减篇幅，可以使用 IDE 自动导入，除非是笔者认为有必要的场景、一些重要的实例它才会出现在代码中。
- 本书的例子大部分使用附录 A 中的数据模型，附录 A 中有基本的论述和对应的 SQL 语句。
- 对于普通的 POJO，笔者大部分都会以 “/\*\*setter and getter\*\*/” 代替 POJO 的 setter 和 getter 方法，类似这样：

```
public class Role {  
    private Long id;  
    private String roleName;  
    private String note;  
    /**setter and getter**/  
}
```

读者可以用 IDE 生成这些属性的 setter 和 getter 方法，这样做主要是为了节省篇幅，突出重点，也有利于读者的阅读。当然在一些特别重要的和使用广泛的场景，比如 MyBatis 入门、SSM 框架整合等场景才会给出全量代码，以便读者进行编码学习。

- 在默认情况下，笔者使用互联网最常用的 MySQL 数据库，当使用其他数据库时，笔者会事先加以说明。
- 本书采用 MyBatis 的版本是 3.4.1，Spring 的版本是 4.3.2，Redis 的版本是 3.2.4，在实践的过程中读者需要注意版本之间的差异。

## 本书的目标读者

阅读本书，读者要掌握以下知识：Java 编程基础、Java EE 基础（JSP、Servlet 等）及数据库基础知识（本书以互联网数据库 MySQL 为主）。本书以互联网企业最广泛使用的技术框架为中心讲解 Java EE 技术，从入门讲解到实践，适合有志于从事 Java EE 开发的各类人员阅读，通过学习本书能够有效提高技术能力，并且将知识点应用到实际的企业工作中去。本书也可以作为大中专院校计算机专业的教材，帮助在校学生学习企业实际应用，当然你也可以把本书当作一本工作手册进行查阅。

## 致谢

本书的成功出版，要感谢电子工业出版社的编辑们，没有他们的辛苦付出，绝对没有本书的成功出版，尤其是编辑汪达文，她启发我创作本书，并且在写作过程中给了我很多的建议和帮助，她为此付出了很多时间和精力。

在撰写本书的过程中，得到了我的师兄周吉文的大力支持，他统稿了全书，也帮助我编写了部分章节的内容；同时还得到梁华辉和谭茂华两位好友的协助，他们以过硬的技术为我排除了不少错误，同时也给了我很多很好的建议，并撰写了一些很好的实例；还要感谢我的姐姐杨坚，她撰写了部分内容，并对书中那些晦涩难懂的句子进行了润色，在此对他们的辛苦付出表示最诚挚的感谢。

互联网技术博大精深，涉及的技术门类特别多，甚至跨行业也特别频繁，技术更新较快。撰写本书时笔者也遇到了一些困难，涉及的知识十分广泛，对技术要求也更高，出错的概率也大大增加，正如没有完美的程序一样，也没有完美的书，一切都需要一个完善的过程，所以尊敬的读者，如果对本书有任何意见或建议，欢迎发送邮件(ykzhen2013@163.com)，或者在博客(<http://blog.csdn.net/ykzhen2015>)上留言，以便于本书的修订。

杨开振

2017年6月

## 读者服务

轻松注册成为博文视点社区用户 ([www.broadview.com.cn](http://www.broadview.com.cn))，扫码直达本书页面。

- **下载资源：**本书如提供示例代码及资源文件，均可在 [下载资源](#) 处下载。
- **提交勘误：**您对书中内容的修改意见可在 [提交勘误](#) 处提交，若被采纳，将获赠博文视点社区积分（在您购买电子书时，积分可用来抵扣相应金额）。
- **交流互动：**在页面下方 [读者评论](#) 处留下您的疑问或观点，与我们和其他读者一同学习交流。

页面入口：<http://www.broadview.com.cn/31847>



# 目 录

## 第 1 部分 入门和技术基础

|                                |    |
|--------------------------------|----|
| 第 1 章 认识 SSM 框架和 Redis         | 2  |
| 1.1 Spring 框架                  | 2  |
| 1.1.1 Spring IoC 简介            | 2  |
| 1.1.2 Spring AOP               | 4  |
| 1.2 MyBatis 简介                 | 6  |
| 1.2.1 Hibernate 简介             | 7  |
| 1.2.2 MyBatis                  | 8  |
| 1.2.3 Hibernate 和 MyBatis 的区别  | 11 |
| 1.3 Spring MVC 简介              | 11 |
| 1.4 最流行的 NoSQL——Redis          | 12 |
| 1.5 SSM+Redis 结构框图及概述          | 13 |
| 第 2 章 Java 设计模式                | 15 |
| 2.1 Java 反射技术                  | 15 |
| 2.1.1 通过反射构建对象                 | 15 |
| 2.1.2 反射方法                     | 17 |
| 2.1.3 实例                       | 18 |
| 2.2 动态代理模式和责任链模式               | 19 |
| 2.2.1 JDK 动态代理                 | 20 |
| 2.2.2 CGLIB 动态代理               | 22 |
| 2.2.3 拦截器                      | 24 |
| 2.2.4 责任链模式                    | 28 |
| 2.3 观察者 (Observer) 模式          | 30 |
| 2.3.1 概述                       | 31 |
| 2.3.2 实例                       | 32 |
| 2.4 工厂模式和抽象工厂模式                | 35 |
| 2.4.1 普通工厂 (Simple Factory) 模式 | 35 |

|                                   |           |
|-----------------------------------|-----------|
| 2.4.2 抽象工厂 (Abstract Factory) 模式  | 36        |
| 2.5 建造者 (Builder) 模式              | 38        |
| 2.5.1 概述                          | 38        |
| 2.5.2 Builder 模式实例                | 39        |
| 2.6 总结                            | 41        |
| <b>第 2 部分 互联网持久框架——MyBatis</b>    |           |
| <b>第 3 章 认识 MyBatis 核心组件</b>      | <b>44</b> |
| 3.1 持久层的概念和 MyBatis 的特点           | 44        |
| 3.2 准备 MyBatis 环境                 | 45        |
| 3.3 MyBatis 的核心组件                 | 46        |
| 3.4 SqlSessionFactory (工厂接口)      | 47        |
| 3.4.1 使用 XML 构建 SqlSessionFactory | 48        |
| 3.4.2 使用代码创建 SqlSessionFactory    | 50        |
| 3.5 SqlSession                    | 50        |
| 3.6 映射器                           | 51        |
| 3.6.1 用 XML 实现映射器                 | 52        |
| 3.6.2 注解实现映射器                     | 53        |
| 3.6.3 SqlSession 发送 SQL           | 54        |
| 3.6.4 用 Mapper 接口发送 SQL           | 55        |
| 3.6.5 对比两种发送 SQL 方式               | 55        |
| 3.7 生命周期                          | 55        |
| 3.7.1 SqlSessionFactoryBuilder    | 56        |
| 3.7.2 SqlSessionFactory           | 56        |
| 3.7.3 SqlSession                  | 56        |
| 3.7.4 Mapper                      | 56        |
| 3.8 实例                            | 57        |
| <b>第 4 章 MyBatis 配置</b>           | <b>63</b> |
| 4.1 概述                            | 63        |
| 4.2 properties 属性                 | 64        |
| 4.2.1 property 子元素                | 64        |
| 4.2.2 使用 properties 文件            | 65        |
| 4.2.3 使用程序传递方式传递参数                | 66        |
| 4.2.4 总结                          | 66        |
| 4.3 settings 设置                   | 66        |
| 4.4 typeAliases 别名                | 69        |
| 4.4.1 系统定义别名                      | 69        |
| 4.4.2 自定义别名                       | 72        |

|              |                            |            |
|--------------|----------------------------|------------|
| 4.5          | typeHandler 类型转换器          | 72         |
| 4.5.1        | 系统定义的 typeHandler          | 73         |
| 4.5.2        | 自定义 typeHandler            | 78         |
| 4.5.3        | 枚举 typeHandler             | 81         |
| 4.5.4        | 文件操作                       | 86         |
| 4.6          | ObjectFactory (对象工厂)       | 87         |
| 4.7          | 插件                         | 89         |
| 4.8          | environments (运行环境)        | 89         |
| 4.8.1        | transactionManager (事务管理器) | 90         |
| 4.8.2        | environment 数据源环境          | 92         |
| 4.9          | databaseIdProvider 数据库厂商标识 | 95         |
| 4.9.1        | 使用系统默认的 databaseIdProvider | 95         |
| 4.9.2        | 不使用系统规则                    | 98         |
| 4.10         | 引入映射器的方法                   | 99         |
| <b>第 5 章</b> | <b>映射器</b>                 | <b>102</b> |
| 5.1          | 概述                         | 102        |
| 5.2          | select 元素——查询语句            | 103        |
| 5.2.1        | 简单的 select 元素的应用           | 104        |
| 5.2.2        | 自动映射和驼峰映射                  | 105        |
| 5.2.3        | 传递多个参数                     | 106        |
| 5.2.4        | 使用 resultMap 映射结果集         | 109        |
| 5.2.5        | 分页参数 RowBounds             | 110        |
| 5.3          | insert 元素——插入语句            | 112        |
| 5.3.1        | 概述                         | 112        |
| 5.3.2        | 简单的 insert 语句的应用           | 113        |
| 5.3.3        | 主键回填                       | 113        |
| 5.3.4        | 自定义主键                      | 114        |
| 5.4          | update 元素和 delete 元素       | 114        |
| 5.5          | sql 元素                     | 115        |
| 5.6          | 参数                         | 116        |
| 5.6.1        | 概述                         | 116        |
| 5.6.2        | 存储过程参数支持                   | 117        |
| 5.6.3        | 特殊字符串的替换和处理 (#和\$)         | 117        |
| 5.7          | resultMap 元素               | 118        |
| 5.7.1        | resultMap 元素的构成            | 118        |
| 5.7.2        | 使用 map 存储结果集               | 119        |
| 5.7.3        | 使用 POJO 存储结果集              | 119        |
| 5.8          | 级联                         | 120        |

|        |                          |     |
|--------|--------------------------|-----|
| 5.8.1  | MyBatis 中的级联             | 121 |
| 5.8.2  | 建立 POJO                  | 124 |
| 5.8.3  | 配置映射文件                   | 127 |
| 5.8.4  | N+1 问题                   | 133 |
| 5.8.5  | 延迟加载                     | 133 |
| 5.8.6  | 另一种级联                    | 137 |
| 5.8.7  | 多对多级联                    | 140 |
| 5.9    | 缓存                       | 143 |
| 5.9.1  | 一级缓存和二级缓存                | 144 |
| 5.9.2  | 缓存配置项、自定义和引用             | 147 |
| 5.10   | 存储过程                     | 149 |
| 5.10.1 | IN 和 OUT 参数存储过程          | 150 |
| 5.10.2 | 游标的使用                    | 152 |
| 第 6 章  | 动态 SQL                   | 155 |
| 6.1    | 概述                       | 155 |
| 6.2    | if 元素                    | 156 |
| 6.3    | choose、when、otherwise 元素 | 156 |
| 6.4    | trim、where、set 元素        | 157 |
| 6.5    | foreach 元素               | 159 |
| 6.6    | 用 test 的属性判断字符串          | 159 |
| 6.7    | bind 元素                  | 160 |
| 第 7 章  | MyBatis 的解析和运行原理         | 162 |
| 7.1    | 构建 SqlSessionFactory 过程  | 163 |
| 7.1.1  | 构建 Configuration         | 165 |
| 7.1.2  | 构建映射器的内部组成               | 165 |
| 7.1.3  | 构建 SqlSessionFactory     | 167 |
| 7.2    | SqlSession 运行过程          | 168 |
| 7.2.1  | 映射器 (Mapper) 的动态代理       | 168 |
| 7.2.2  | SqlSession 下的四大对象        | 172 |
| 7.2.3  | SqlSession 运行总结          | 179 |
| 第 8 章  | 插件                       | 181 |
| 8.1    | 插件接口                     | 181 |
| 8.2    | 插件的初始化                   | 182 |
| 8.3    | 插件的代理和反射设计               | 183 |
| 8.4    | 常用的工具类——MetaObject       | 186 |
| 8.5    | 插件开发过程和实例                | 187 |
| 8.5.1  | 确定需要拦截的签名                | 187 |

|       |            |     |
|-------|------------|-----|
| 8.5.2 | 实现拦截方法     | 189 |
| 8.5.3 | 配置和运行      | 191 |
| 8.5.4 | 插件实例——分页插件 | 192 |
| 8.6   | 总结         | 205 |

## 第 3 部分 Spring 基础

|        |                                 |     |
|--------|---------------------------------|-----|
| 第 9 章  | Spring IoC 的概念                  | 208 |
| 9.1    | Spring 的概述                      | 208 |
| 9.2    | Spring IoC 概述                   | 210 |
| 9.2.1  | 主动创建对象                          | 211 |
| 9.2.2  | 被动创建对象                          | 213 |
| 9.2.3  | Spring IoC 阐述                   | 214 |
| 9.3    | Spring IoC 容器                   | 215 |
| 9.3.1  | Spring IoC 容器的设计                | 215 |
| 9.3.2  | Spring IoC 容器的初始化和依赖注入          | 218 |
| 9.3.3  | Spring Bean 的生命周期               | 218 |
| 9.4    | 小结                              | 223 |
| 第 10 章 | 装配 Spring Bean                  | 224 |
| 10.1   | 依赖注入的 3 种方式                     | 224 |
| 10.1.1 | 构造器注入                           | 224 |
| 10.1.2 | 使用 setter 注入                    | 225 |
| 10.1.3 | 接口注入                            | 226 |
| 10.2   | 装配 Bean 概述                      | 227 |
| 10.3   | 通过 XML 配置装配 Bean                | 228 |
| 10.3.1 | 装配简易值                           | 228 |
| 10.3.2 | 装配集合                            | 229 |
| 10.3.3 | 命名空间装配                          | 233 |
| 10.4   | 通过注解装配 Bean                     | 235 |
| 10.4.1 | 使用@Component 装配 Bean            | 236 |
| 10.4.2 | 自动装配——@Autowired                | 239 |
| 10.4.3 | 自动装配的歧义性 (@Primary 和@Qualifier) | 241 |
| 10.4.4 | 装载带有参数的构造方法类                    | 244 |
| 10.4.5 | 使用@Bean 装配 Bean                 | 245 |
| 10.4.6 | 注解自定义 Bean 的初始化和销毁方法            | 245 |
| 10.5   | 装配的混合使用                         | 246 |
| 10.6   | 使用 Profile                      | 249 |
| 10.6.1 | 使用注解@Profile 配置                 | 249 |
| 10.6.2 | 使用 XML 定义 Profile               | 250 |

|         |                             |     |
|---------|-----------------------------|-----|
| 10.6.3  | 启动 Profile                  | 252 |
| 10.7    | 加载属性 (properties) 文件        | 254 |
| 10.7.1  | 使用注解方式加载属性文件                | 254 |
| 10.7.2  | 使用 XML 方式加载属性文件             | 257 |
| 10.8    | 条件化装配 Bean                  | 258 |
| 10.9    | Bean 的作用域                   | 259 |
| 10.10   | 使用 Spring 表达式 (Spring EL)   | 261 |
| 10.10.1 | Spring EL 相关的类              | 261 |
| 10.10.2 | Bean 的属性和方法                 | 264 |
| 10.10.3 | 使用类的静态常量和方法                 | 265 |
| 10.10.4 | Spring EL 运算                | 265 |
| 第 11 章  | 面向切面编程                      | 267 |
| 11.1    | 一个简单的约定游戏                   | 267 |
| 11.1.1  | 约定规则                        | 267 |
| 11.1.2  | 读者的代码                       | 269 |
| 11.1.3  | 笔者的代码                       | 271 |
| 11.2    | Spring AOP 的基本概念            | 274 |
| 11.2.1  | AOP 的概念和使用原因                | 274 |
| 11.2.2  | 面向切面编程的术语                   | 278 |
| 11.2.3  | Spring 对 AOP 的支持            | 280 |
| 11.3    | 使用 @AspectJ 注解开发 Spring AOP | 280 |
| 11.3.1  | 选择连接点                       | 281 |
| 11.3.2  | 创建切面                        | 281 |
| 11.3.3  | 定义切点                        | 283 |
| 11.3.4  | 测试 AOP                      | 285 |
| 11.3.5  | 环绕通知                        | 287 |
| 11.3.6  | 织入                          | 289 |
| 11.3.7  | 给通知传递参数                     | 289 |
| 11.3.8  | 引入                          | 290 |
| 11.4    | 使用 XML 配置开发 Spring AOP      | 293 |
| 11.4.1  | 前置通知、后置通知、返回通知和异常通知         | 294 |
| 11.4.2  | 环绕通知                        | 296 |
| 11.4.3  | 给通知传递参数                     | 297 |
| 11.4.4  | 引入                          | 298 |
| 11.5    | 经典 Spring AOP 应用程序          | 299 |
| 11.6    | 多个切面                        | 301 |
| 11.7    | 小结                          | 306 |



|                                   |     |
|-----------------------------------|-----|
| 第 12 章 Spring 和数据库编程              | 307 |
| 12.1 传统的 JDBC 代码的弊端               | 307 |
| 12.2 配置数据库资源                      | 309 |
| 12.2.1 使用简单数据库配置                  | 309 |
| 12.2.2 使用第三方数据库连接池                | 310 |
| 12.2.3 使用 JNDI 数据库连接池             | 310 |
| 12.3 JDBC 代码失控的解决方案——jdbcTemplate | 311 |
| 12.3.1 jdbcTemplate 的增、删、查、改      | 312 |
| 12.3.2 执行多条 SQL                   | 314 |
| 12.3.3 jdbcTemplate 的源码分析         | 315 |
| 12.4 MyBatis-Spring 项目            | 317 |
| 12.4.1 配置 SqlSessionFactoryBean   | 318 |
| 12.4.2 SqlSessionTemplate 组件      | 322 |
| 12.4.3 配置 MapperFactoryBean       | 324 |
| 12.4.4 配置 MapperScannerConfigurer | 324 |
| 12.4.5 测试 Spring+MyBatis          | 327 |
| 第 13 章 深入 Spring 数据库事务管理          | 330 |
| 13.1 Spring 数据库事务管理器的设计           | 331 |
| 13.1.1 配置事务管理器                    | 333 |
| 13.1.2 用 Java 配置方式实现 Spring 数据库事务 | 334 |
| 13.2 编程式事务                        | 336 |
| 13.3 声明式事务                        | 337 |
| 13.3.1 Transactional 的配置项         | 337 |
| 13.3.2 使用 XML 进行配置事务管理器           | 339 |
| 13.3.3 事务定义器                      | 340 |
| 13.3.4 声明式事务的约定流程                 | 341 |
| 13.4 数据库的相关知识                     | 343 |
| 13.4.1 数据库事务 ACID 特性              | 343 |
| 13.4.2 丢失更新                       | 343 |
| 13.4.3 隔离级别                       | 344 |
| 13.5 选择隔离级别和传播行为                  | 347 |
| 13.5.1 选择隔离级别                     | 347 |
| 13.5.2 传播行为                       | 348 |
| 13.6 在 Spring+MyBatis 组合中使用事务     | 350 |
| 13.7 @Transactional 的自调用失效问题      | 358 |
| 13.8 典型错误用法的剖析                    | 363 |
| 13.8.1 错误使用 Service               | 363 |
| 13.8.2 过长时间占用事务                   | 364 |