Broadview®
www.broadview.com.cn

媲美于SSH组合的轻量级Java EE开发方式

Spring+MyBatis
企业应用实战

疯狂软件 编著

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Spring+MyBatis
企业应用实战

疯狂软件　编著

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

内 容 简 介

本书介绍了 Java EE 领域的两个开源框架：Spring 的 MVC 和 MyBatis。其中 Spring 的版本为 4.2，MyBatis 的版本是 3.4。本书的示例建议在 Tomcat 8 上运行。

本书重点介绍如何整合 Spring MVC 4+MyBatis 3 进行 Java EE 开发，主要包括三个部分。第一部分详细介绍了 Spring MVC 框架的用法。第二部分详细介绍了 MyBatis 框架的用法。第三部分重点介绍了 Spring MVC 4 ＋MyBatis 3 的整合，并示范开发了一个包含 6 个表、表之间具有复杂的关联映射关系，且业务功能也相对完善的 HRM 人事管理系统案例，希望让读者理论联系实际，将这两个框架真正运用到实际开发当中去。该案例采用目前最流行、最规范的 Java EE 架构，整个应用分为 DAO 持久层、领域对象层、业务逻辑层、控制器层和视图层，各层之间分层清晰，层与层之间以松耦合的方法组织在一起。所有代码完全基于 Eclipse IDE 来完成，一步步带领读者深入两个框架的核心。

阅读本书之前，建议先阅读疯狂软件教育的《疯狂 Java 讲义》一书。本书适合有较好的 Java 编程基础，JSP、Servlet、JDBC 基础，Spring 框架基础的读者，尤其适合于对 Spring MVC 和 MyBatis 了解不够深入，或对 Spring MVC+MyBatis 整合开发不太熟悉的开发人员阅读。

图书在版编目（CIP）数据

Spring+MyBatis 企业应用实战 / 疯狂软件编著. —北京：电子工业出版社，2017.1
ISBN 978-7-121-30421-7

Ⅰ. ①S… Ⅱ. ①疯… Ⅲ. ①JAVA 语言－程序设计 Ⅳ. ①TP312.8

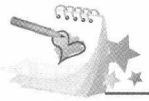
中国版本图书馆 CIP 数据核字（2016）第 284499 号

责任编辑：张月萍
特约编辑：梁卫红
印　　刷：涿州市京南印刷厂
装　　订：涿州市京南印刷厂
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：20.5　　字数：543 千字
版　　次：2017 年 1 月第 1 版
印　　次：2017 年 3 月第 3 次印刷
印　　数：8501~14500 册　　定价：58.00 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。



前　言

时至今日，以 Spring 为核心的轻量级 Java EE 企业开发平台在企业开发中占有绝对的优势，Java EE 应用以其稳定的性能、良好的开放性以及严格的安全性，深受企业应用开发者的青睐，应用的性能、稳定性都有很好的保证。

轻量级 Java EE 开发大致可分为两种方式：以 Struts+Spring+Hibernate 三大框架为核心的轻量级 Java EE，以 Spring MVC+MyBatis 为核心的轻量级 Java EE。这两种组合都在保留经典 Java EE 应用架构、高度可扩展性、高度可维护性的基础上，降低了 JavaEE 应用的开发、部署成本，对于大部分中小型企业应用是第一首选。

本书重点介绍以 Spring MVC+MyBatis 为核心的轻量级 Java EE。本书采用 Tomcat 8 作为 Web 服务器，Eclipse IDE 作为开发工具，详细介绍了 Spring MVC 和 MyBatis 框架。Spring MVC 的配置全部使用注解方式，不再采用旧版本中传统的 XML 配置方式；MyBatis 则基于企业开发实际情况，首先介绍了传统的 XML 持久化映射，之后又介绍了升级的注解持久化映射。

随着 Spring 的不断发展，Spring MVC 已有取代 Struts 的能力，而 MyBatis 相对 Hibernate 而言则更为轻便、简单，越来越多的公司选择使用 Spring MVC+MyBatis 的轻量级框架组合来开发 Java EE 项目，掌握 Spring MVC+MyBatis 的技术将为 Java 开发者带来更多的就业机会与竞争力。

本书有什么特点

本书是一本介绍 Spring MVC+MyBatis 技术以及两者整合的实用图书，全面介绍了最新的 Spring MVC 和 MyBatis 各方面的知识。

本书针对每一个知识点都通过相应的程序给出了示范，第 14 章的实战项目“HRM 人事管理系统”采用目前最流行、最规范的 Java EE 架构，整个应用分为 DAO 持久层、领域对象层、业务逻辑层、控制器层和视图层，各层之间分层清晰，层与层之间以松耦合的方法组织在一起。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，现如今从事专业、高端的职业技术培训，所有应用范例都密切契合企业开发实际场景，例如用户权限验证、文件上传下载等都是企业开发实际功能，同时采用了目前企业最流行、最规范的开发架构，严格遵守 Java EE 开发规范。读者参考本书的架构，完全可以身临其境地感受企业实际项目开发。

本书并不是一本关于所谓“思想”的书，也没有一堆“深奥”的新名词和“高深”的思想，只会让读者学会实际的 Spring MVC 和 MyBatis 技术。本书的特点是操作步骤详细，编程思路清晰，语言平实易懂。只要读者认真阅读本书，并掌握书中知识，那么就完全可以胜任企业中 Spring MVC+MyBatis 项目开发。

阅读本书需要具备一定的计算机知识以及编程功底。熟练掌握 Java 语言和 Spring 框架的 IOC、AOP 和持久层的 ORM 设计模式等知识对于学习本书是很有必要的。

可访问 www.crazyit.org 或 www.broadview.com.cn/30421 下载本书配套资源。

本书写给谁看

如果你已经掌握 Java SE 内容，或已经学完了疯狂软件教育的《疯狂 Java 讲义》一书，那么你非常适合阅读此书。除此之外，如果你已有初步的 JSP、Servlet、JDBC 基础，甚至对 Spirng、MyBatis 等框架有所了解，但希望掌握它们在实际开发中应用，本书也将非常适合你。如果你对 Java 的掌握还不熟练，则建议遵从学习规律，循序渐进，暂时不要购买、阅读此书，而是按照“疯狂 Java 学习路线图”中的建议顺序学习。

衷心感谢

衷心感谢李刚老师，他是一个非常好的朋友，在本书的创作过程中，他提供了大量切实、有用的帮助。同时衷心感谢疯狂软件教育中心所有同事提供的帮助。

感谢所有参加疯狂软件实训的学生，他们在实际工作场景的应用证明了本书的价值，他们的反馈让本书更加实用。

肖文吉

2016 年 11 月 16 日

目 录 CONTENTS

CHAPTER

1

第 1 章
Java EE 应用

本章要点

- Java EE 应用的基础知识
- Java EE 应用的模型和相关组件
- Java EE 应用的结构和优势
- 轻量级 Java EE 应用的相关技术

时至今日，轻量级 Java EE 平台在企业开发中占有绝对的优势，Java EE 应用以其稳定的性能、良好的开放性以及严格的安全性，深受企业应用开发者的青睐。实际上，对于信息化要求较高的行业，如银行、电信、证券以及电子商务等，都不约而同地选择了 Java EE 作为开发平台。

对于一个企业而言，选择 Java EE 构建信息化平台，更体现了一种长远的规划：企业的信息化是不断整合的过程，在未来的日子里，经常会有不同平台、不同的异构系统需要整合。Java EE 应用提供的跨平台、开放性以及各种远程访问技术，为异构系统的良好整合提供了保证。

一些有高并发、高稳定要求的电商网站（如淘宝、京东等），公司创立之初并没有采用 Java EE 技术架构（淘宝早期用 PHP，京东早期用.NET），但当公司的业务一旦真正开始，他们马上就发现 PHP、.NET 无法支撑公司业务运营，后来全部改为使用 Java EE 技术架构。就目前的局面来看，Java EE 已经成为真正企业级应用的不二之选。

1.1 Java EE 应用概述

今天所说的 Java EE 应用，超出了 Sun 所提出的经典 Java EE 应用规范，而是一种更广泛的开发规范。经典 Java EE 应用往往以 EJB（企业级 Java Bean）为核心，以应用服务器为运行环境，所以开发、运行成本较高。本书所介绍的 Spring MVC + MyBatis 作为轻量级 Java EE 应用不仅具备 Java EE 规范的种种特征，例如面向对象建模的思维方式、优秀的应用分层及良好的可扩展性、可维护性，而且保留了经典 Java EE 应用的架构，但其开发、运行成本更低。

1.1.1 Java EE 应用的分层模型

不管是经典的 Java EE 架构，还是本书介绍的轻量级 Java EE 架构，大致上都可分为如下几层：

- **Domain Object（领域对象）层。**此层由一系列的 POJO（Plain Old Java Object，普通的、传统的 Java 对象）组成，这些对象是该系统的 Domain Object（领域对象），往往包含了各自所需实现的业务逻辑方法。
- **DAO（Data Access Object，数据访问对象）层。**此层由一系列的 DAO 组件组成，这些 DAO 实现了对数据库的创建、查询、更新和删除（CRUD）等原子操作。

注意：

在经典 Java EE 应用中，DAO 层也被称为 EAO 层，EAO 层组件的作用与 DAO 层组件的作用基本相似。只是 EAO 层主要完成对实体（Entity）的 CRUD 操作，因此简称为 EAO 层。

DAO 层在 MyBatis 中也被称为 Mapper 层，其通过 SQL 语句的映射完成 CRUD 操作。

- **Service（业务逻辑）层。**此层由一系列的业务逻辑对象组成，这些业务逻辑对象实现了系统所需要的业务逻辑方法。这些业务逻辑方法可能仅仅用于暴露 Domain Object 对象所实现的业务逻辑方法，也可能是依赖 DAO 组件实现的业务逻辑方法。
- **Controller（控制器）层。**此层由一系列控制器组成，这些控制器用于拦截用户请求，并调用业务逻辑组件的业务逻辑方法，处理用户请求，并根据处理结果向不同的表现层组件转发。

- View（表现）层。此层由一系列的 JSP 页面、Velocity 页面、PDF 文档视图组件组成，负责收集用户请求，并显示处理结果。

大致上，Java EE 应用的架构如图 1.1 所示。

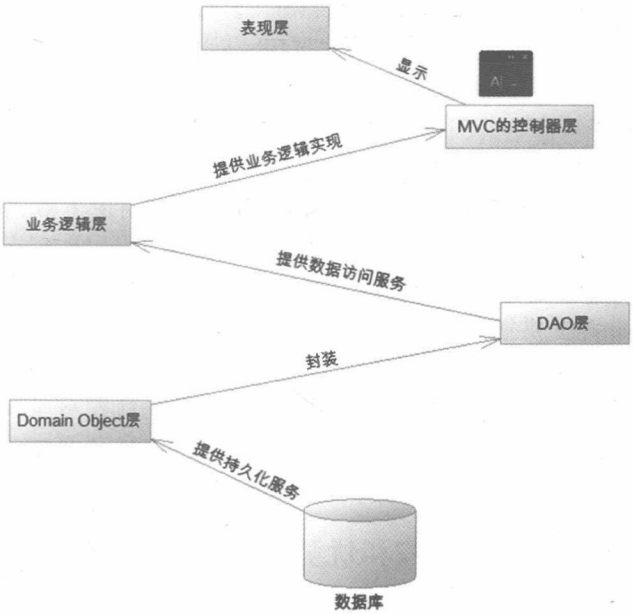


图 1.1 Java EE 应用的架构

各层的 Java EE 组件之间以松耦合的方式耦合在一起，各组件并不以硬编码方式耦合，这种方式是为了应用以后的扩展性。从上向下，上面组件的实现依赖于下面组件的功能；从下向上，下面组件支持上面组件的实现。

1.1.2 Java EE 应用的组件

通过上一节的介绍，我们可以看到 Java EE 应用提供了系统架构上的飞跃。Java EE 架构提供了良好的分离，隔离了各组件之间的代码依赖。

总体而言，Java EE 应用大致包括如下几类组件：

- **表现层组件**。主要负责收集用户输入数据，或者向客户显示系统状态。最常用的表现层技术是 JSP，但 JSP 并不是唯一的表现层技术。表现层还可由 Velocity、FreeMarker 和 Tapestry 等技术完成，或者使用普通的应用程序充当表现层组件，甚至可以是小型智能设备。
- **控制器组件**。对于 Java EE 的 MVC 框架，其提供一个前端核心控制器，核心控制器负责拦截用户请求，并将请求转发给用户实现的控制器组件。这些用户实现的控制器组件则负责调用业务逻辑方法，处理用户请求。
- **业务逻辑组件**。这是系统的核心组件，实现系统的业务逻辑。通常，一个业务逻辑方法对应一次用户操作。一个业务逻辑方法应该是一个整体，因此要求对业务逻辑方法增加事务性。业务逻辑方法仅仅负责实现业务逻辑，不应该进行数据库访问。因此，业务逻辑组件中不应该出现原始的 MyBatis、Hibernate 和 JDBC 等 API。

提示：

保证业务逻辑组件之中不出现 MyBatis、Hibernate 和 JDB 等 API，有一个更重要的原因：为保证业务逻辑方法的实现，与具体的持久层访问技术分离。当系统

需要在不同持久层技术之间切换时，系统的业务逻辑组件无须任何改变。有时会见到一些所谓的 Java EE 应用，居然在 JSP 页面里面调用 SqlSessionFactory、SqlSession 等 API，这无疑是非常荒唐的，这种应用仅仅是使用 MyBatis，完全没有脱离 Model 1 的 JSP 开发模式，这是相当失败的结构。实际上，不仅 JSP，Servlet 中也不应出现持久层 API，包括 JDBC、MyBatis、Hibernate API。最理想的情况是，业务逻辑组件中都不应出现持久层 API。

- **DAO 组件**。这个类型的对象比较缺乏变化，每个 DAO 组件都提供 Domain Object 对象基本的创建、查询、更新和删除等操作，这些操作对应于数据库的 CRUD（创建、查询、更新和删除）等原子操作。当然，如果采用不同的持久层访问技术，DAO 组件的实现会完全不同。为了业务逻辑组件的实现与 DAO 组件的实现分离，程序应该为每个 DAO 组件都提供接口，业务逻辑组件面向 DAO 接口编程，这样才能提供更好的解耦。
- **领域对象组件**。领域对象（Domain Object）抽象了系统的对象模型。通常而言，这些领域对象的状态都必须保存在数据库里。因此，每个领域对象通常对应一个或多个数据表，领域对象通常需要提供对数据记录的访问方式。

1.1.3 Java EE 应用的结构和优势

作为 Java EE 的初学者，常常有一个问题：明明可以使用 JSP 完成这个系统，为什么还要使用 MyBatis 和 Hibernate 等技术？难道仅仅是为了听起来高深一些？明明可以使用纯粹的 JSP 完成整个系统，为什么还要将系统分层？

要回答这些问题，就不能仅仅考虑系统开发过程，还需要考虑系统后期的维护、扩展；而且不能仅仅考虑小型系统，还要考虑大型系统的协同开发。如果是用于个人学习、娱乐的个人站点，的确没有必要使用复杂的 Java EE 应用架构，采用纯粹的 JSP 就可以实现整个系统。

但对于大型的信息化系统，采用 Java EE 应用架构则有很大的优势。

对于信息化系统，前期开发工作对整个系统工作量而言，仅仅是小部分，而后期的维护、升级往往占更大的比重。更极端的情况是，可能在前期开发期间，企业需求已经发生变化，而这种改变是客观的，软件系统必须适应这种改变，这要求软件系统具有很好的伸缩性。

这种框架结构其目的是让应用的各组件以松耦合的方式组织在一起，让应用之间的耦合停留在接口层次，而不是代码层次。

1.2 轻量级 Java EE 应用相关技术

轻量级 Java EE 应用以传统的 JSP 作为表现层技术，以一系列开源框架作为 MVC 层、中间层、持久层解决方案，并将这些开源框架有机地组合在一起，使得 Java EE 应用具有高度的可扩展性、可维护性。

1.2.1 JSP、Servlet 和 JavaBean 及替代技术

JSP 是最早的 Java EE 规范之一，也是最经典的 Java EE 技术之一。直到今天，JSP 依然广泛地应用于各种 Java EE 应用中，充当 Java EE 应用的表现层角色。

JSP 具有简单、易用的特点，JSP 的学习路线平坦，而且国内有大量 JSP 学习资料，所以大部分 Java 学习者学习 Java EE 开发都会选择从 JSP 开始。

Servlet 和 JSP 其实是完全统一的，二者底层的运行原理是完全一样的。实际上，JSP 必须被 Web 服务器编译成 Servlet，真正在 Web 服务器内运行的是 Servlet。从这个意义上来看，JSP 相当于一个“草稿”文件，Web 服务器根据该“草稿”文件生成 Servlet，真正提供 HTTP 服务的是 Servlet，因此广义的 Servlet 包含了 JSP 和 Servlet。

就目前的 Java EE 应用来看，纯粹的 Servlet 已经很少使用了，毕竟 Servlet 的开发成本太高，而且使用 Servlet 充当表现层将导致表现层页面难以维护，不利于美工人员参与 Servlet 开发，所以在实际开发中大都使用 JSP 充当表现层技术。

Servlet3.x 规范的出现，再次为 Java Web 开发带来了巨大的便捷。Servlet3.x 提供了异步请求、注解、增强的 Servlet API、非阻塞 IO 功能，这些功能都极大地简化了 Java Web 开发。

由于 JSP 只负责简单的显示逻辑，因此 JSP 无法直接访问应用的底层状态，Java EE 应用会选择使用 JavaBean 来传输数据。在严格的 Java EE 应用中，中间层的组件会将应用底层的状态信息封装成 JavaBean 集，这些 JavaBean 也被称为 DTO（Data Transfer Object，数据传输对象），并将这些 DTO 集传到 JSP 页面，从而让 JSP 可以显示应用的底层状态。

在目前阶段，Java EE 应用除了可以使用 JSP 作为表现层技术之外，还可以使用 FreeMarker 或 Velocity 作为表现层技术，这些表现层技术更加纯粹，使用起来更加便捷，完全可作为 JSP 的替代。

1.2.2 MyBatis3 及替代技术

传统的 Java 应用都是采用 JDBC 来访问数据库的，但传统的 JDBC 采用的是一种基于 SQL 的操作方式，这种操作方式与 Java 语言的面向对象特性不太一致，所以 Java EE 应用需要一种技术，通过这种技术能让 Java 以面向对象的方式操作关系数据库。

这种特殊的技术就是 ORM（Object Relation Mapping），最早的 ORM 是 Entity EJ（Enterprise JavaBean），EJB 就是经典 Java EE 应用的核心，从 EJB1.0 到 EJB 2.x，许多人会觉得 EJB 非常烦琐，所以导致 EJB 备受诟病。

在这种背景下，Hibernate 框架应运而生。Hibernate 框架是一种开源的、轻量级的 ORM 框架，它允许将普通的、传统的 Java 对象（POJO）映射成持久化类，允许应用程序以面向对象的方式来操作 POJO，而 Hibernate 框架则负责将这种操作转换成底层的 SQL 操作。

大多数情况下（特别是对新项目、新系统的开发而言），Hibernate 这样的机制无往不利，大有一统天下的势头。但是，在一些特定的环境下，Hibernate 这种一站式的解决方案却未必适合。如：

- 系统的部分或全部数据来自现有数据库，出于安全考虑，只对开发团队提供几条 Select SQL（或存储过程）以获取所需数据，具体的表结构不予公开。
- 开发规范中要求，所有牵涉到业务逻辑部分的数据库操作，必须在数据库层由存储过程实现（就金融行业而言，工商银行、中国银行、交通银行等商业银行都曾在开发规范中严格指定）。
- 系统数据处理量巨大，性能要求极为苛刻，这往往意味着我们必须通过经过高度优化的 SQL 语句（或存储过程）才能达到系统性能设计指标。

面对这样的需求，Hibernate 不再适合，甚至无法使用。此时，直接使用 JDBC 进行数据库操作实际上也是不错的选择，只是拖沓的数据库访问代码、乏味的字段读取操作令人厌烦，而“半自动化”的 MyBatis，却正好解决了这个问题。

这里的“半自动化”，是相对 Hibernate 等提供了全面的数据库封装机制的“全自动化”ORM 实现而言的，“全自动”ORM 实现了 POJO 和数据库表之间的映射，以及 SQL 的自

动生成和执行。而 MyBatis 的着力点，则在于 POJO 与 SQL 之间的映射关系。也就是说，使用 MyBatis 提供的 ORM 机制，对业务逻辑实现人员而言，面对的是纯粹的 Java 对象，这一层与通过 Hibernate 实现 ORM 而言基本一致，而对于具体的数据操作，Hibernate 会自动生成 SQL 语句，但 MyBatis 则并不会为程序员在运行期自动生成 SQL 语句。具体的 SQL 需要程序员编写，然后通过映射配置文件，将 SQL 所需的参数以及返回的结果字段映射到指定的 POJO。

相对 Hibernate 等“全自动”ORM 机制而言，MyBatis 以 SQL 开发的工作量和数据库移植性上的让步，为系统设计提供了更大的自由空间。作为“全自动”ORM 实现的一种有益补充，MyBatis 的存在具有特别的意义。

MyBatis 是 Apache 组织提供的一个轻量级持久层框架，是一个支持普通 SQL 查询、存储过程和高级映射的优秀持久层框架。MyBatis 消除了几乎所有的 JDBC 代码和参数的手工设置过程以及对结果集的检索封装。MyBatis 可以使用简单的 XML 或注解来进行配置和原始映射，将接口和 Java 的 POJO 映射成数据库中的记录。

MyBatis 作为持久层框架，其主要思想是将程序中的大量 SQL 语句剥离出来，配置在配置文件中，实现 SQL 的灵活配置。这样做的好处是将 SQL 与程序代码分离，可以在不修改程序代码的情况下，直接在配置文件中修改 SQL。

MyBatis 最新版本是 3.4.1，这也是本书所使用的 MyBatis 版本。

除此之外，Oracle 的 TopLink、Apache 的 OJB 都可作为替代方案。但由于种种原因，它们并未得到广泛的市场支持，所以这两个框架的资料、文档相对比较少，选择它们需要一定的勇气和技术功底。

1.2.3 Spring4 及替代技术

如果你有 5 年以上的 Java EE 开发经验，并主持过一些大型项目的设计，你会发现 Spring 框架似曾相识。Spring 甚至没有太多的新东西，它只是抽象了大量 Java EE 应用中的常用代码，将它们抽象成一个框架。通过使用 Spring 可以大幅度地提高开发效率，并可以保证整个应用具有良好的设计。

Spring 框架里充满了各种设计模式的应用，如单例模式、工厂模式、抽象工厂模式、命令模式、职责链模式、代理模式等，Spring 框架的用法、源码则更是一道丰盛的 Java 大餐。

Spring 框架号称 Java EE 应用的一站式解决方案，Spring 本身提供了一个设计优良的 MVC 框架：Spring MVC。使用 Spring 框架可以直接使用该 MVC 框架。由于 Spring 框架拥有极高的市场占有率，因此越来越多的 Spring 框架的使用者使用 Spring MVC 替代曾经的 MVC 框架的王者 Struts2。当然，Spring 也可以无缝地整合 Struts2、JSF 等优秀的 MVC 框架。

Spirng 框架并未提供完整的持久层框架，可以将其理解成一种“空”，但这种“空”正是 Spring 框架的魅力所在。Spring 能与大部分持久层框架无缝整合：MyBatis、Hibernate、JPA、TopLink，更甚至直接使用 JDBC，随便你喜欢，无论选择哪种持久层框架，Sping 都会为你提供无缝的整合及极好的简化。

从这个意义上来看，Spring 更像一种中间层容器，Spring 向上可以与 MVC 框架无缝整合，向下可以与各种持久层框架无缝整合，其的确具有强大的生命力。由于 Spring 框架的特殊地位，因而轻量级 Java EE 应用通常都不会拒绝使用 Spring。实际上，轻量级 Java EE 这个概念也是由 Spring 框架衍生出来的。Spring 框架暂时没有较好的替代框架。

Spring 的最新版本是 4.2.0，本书所介绍的 Spring 也是基于该版本。

1.2.4 使用开源框架的好处

以上提到的 Struts2、MyBatis3、Hibernate4、Spring4 等都是 Java 领域最常见的框架，这些框架得到广泛的开发者支持，它们能极大地提高 Java EE 应用的开发效率，并能保证应用具有稳定的性能。

越来越多的企业开始选择 Spring MVC+MyBatis 来构建系统架构，在电商热门的今天，Spring MVC+MyBatis 已成为电商项目架构的最佳搭配。本书将重点讲解 Spring MVC+ MyBatis 如何无缝整合开发 Java EE 项目。

常常有些初学者，甚至包括一些所谓的企业开发人士提出：为什么需要使用框架？用 JSP 和 Servlet 已经足够了。

提出这些疑问的人通常还未真正进入企业开发，或者从未开发一个真正的项目。因为真实的企业应用开发有两个重要的关注点：可维护性和复用。

先从软件的可维护性来考虑这种说法。全部采用 JSP 和 Servlet 的应用，因为分层不够清晰，业务逻辑的实现没有单独分离出来，从而造成系统后期维护困难。

从软件复用角度来考虑。这是一个企业开发的生命，企业以追求利润为最大目标，企业希望以最快的速度，开发出最稳定、最实用的软件。因为系统没有使用任何框架，每次开发系统都需要重新开发，重新开发的代码具有更多的漏洞，这就增加了系统出错的风险；另外，每次开发新代码都需要投入更多的人力和物力。

以笔者多年的实际开发经验来看，每个公司都会有自己的基础类库，这就是软件的复用，这些基础类库将在后续开发中多次被重复使用。例如，信息化系统，其中总有一些开发过程是重复的，为什么不将这些重复开发工作抽象成基础类库呢？这种抽象既提高了开发效率，而且因为重复使用，也降低了引入错误的风险。

因此只要是一个有实际开发经验的软件公司，就一定有自己的一套基础类库，这就是需要使用框架的原因。从某个角度来看，框架也是一套基础类库，它抽象了软件开发的通用步骤，让实际开发人员可以直接利用这部分实现。当然，即使使用 JSP 和 Servlet 开发的公司，也可以抽象出自己的一套基础类库，那么这也是框架！一个从事实际开发的软件公司，不管它是否意识到，它已经在使用框架。区别只有：使用的框架到底是别人提供的，还是自己抽象出来的。

到底是使用第三方提供的框架更好，还是使用自己抽象的框架更好？这个问题就见仁见智了。通常而言，第三方提供的框架更稳定，更有保证，因为第三方提供的框架往往经过了更多人的测试。而使用自己抽象的框架，则更加熟悉底层运行原理，在处理问题上更有方向性。如果不是有非常特殊的理由，还是推荐使用第三方框架，特别是那些流行的、广泛使用的、开源的框架。

1.3 本章小结

本章主要介绍了 Java EE 应用的相关基础知识，其中，简要介绍了 Java EE 应用应该遵循怎样的架构模型，通常应该具有哪些组件，以及这些组件通常使用什么样的技术来实现。本章还简单归纳了 Java EE 应用所具有的优势和吸引力。

本书使用的是 Apache Tomcat Web 服务器，使用的开发工具是 Eclipse。关于 Tomcat 的安装和 Eclipse 工具的具体用法，请参考“疯狂 Java 系列”之《轻量级 Java EE 企业应用实战》，这里不做讨论。

第 2 章将重点介绍 Spring MVC。

CHAPTER

2

第2章 Spring MVC 简介

本章要点

- Model1 和 Model2
- MVC 思想及其优点
- Spring MVC 的优势
- Spring MVC 的前端控制器 DispatcherServlet
- 开发第一个 Spring MVC 应用
- 基于 Controller 接口的控制器
- 基于注解的控制器
- Spring MVC 的工作流程