“十二五”科学技术专著丛书

Java Web 核心技术

毋建军 著

CORE JAVA WEB
PROGRAMMING
TUTORIALS

北京邮电大学出版社
www.buptpress.com

“十二五”科学技术专著丛书

Java Web 核心技术

毋建军 著

北京邮电大学出版社
www. buptpress. com

内 容 简 介

Java Web 应用已经成为当前互联网主流的应用发展方向之一，尤以移动互联网为代表的领域，更是离不开 Java Web 技术的支撑，如何快速地从繁多、杂乱的 Web 技术中学习核心基础内容，是许多学习者面临的技术门槛，本书作者希望通过理论和实际应用相结合的方式，让初入者能够理解其关注的技术内容。

本书作为 Java Web 原理与技术应用的入门书籍，内容全面且通俗易懂，对 Java Web 应用所涉及的关键核心技术进行了全面的详解，除了对 Web 基础技术 HTML、JavaScript、Servlet、JSP 解析之外，还详细讲述了核心技术应用开发中的设计模式、框架模式、Web 应用服务器、JDBC、XML、Struts、Hibernate、Spring 等技术及实践应用，以帮助读者学习和深入理解相关技术。

本书适用于对 Java Web 应用技术感兴趣的初学者、技术人员，可作为大中专院校软件开发、移动应用开发相关专业的教材，也可作为 Java Web 开发人员的参考书。

图书在版编目(CIP)数据

Java Web 核心技术 / 毋建军著. -- 北京 : 北京邮电大学出版社，2015.5
ISBN 978-7-5635-4301-4

Ⅰ. ①J… Ⅱ. ①毋… Ⅲ. ①JAVA 语言－程序设计 Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字（2015）第 033713 号

书　　名：Java Web 核心技术
著作责任者：毋建军　著
责 任 编 辑：徐振华　孙宏颖
出 版 发 行：北京邮电大学出版社
社　　址：北京市海淀区西土城路 10 号（邮编：100876）
发 行 部：电话：010-62282185　传真：010-62283578
E-mail：publish@bupt.edu.cn
经　　销：各地新华书店
印　　刷：北京源海印刷有限责任公司
开　　本：787 mm×1 092 mm　1/16
印　　张：19.5
字　　数：507 千字
版　　次：2015 年 5 月第 1 版　2015 年 5 月第 1 次印刷

ISBN 978-7-5635-4301-4　　定　价：42.00 元

前　言

近年来，随着互联网行业技术的不断成熟和发展，传统的软件开发基础技术已经远远不能满足当前社会的需求，无论是以淘宝、京东为代表的电子商务网站应用，还是以移动互联网为基础的移动应用，都不再是仅靠原有的C和Java基础语言就能实现Web方面的应用，以C系列(C、C++、C#)和Java系列为主的两大技术阵营，各自都推出了基于Web的应用开发技术，围绕Visual Studio Net集成开发平台为代表的C系列Web开发和MyEclipse集成开发平台为代表的Java系列Web开发，催生了许多关键技术，特别是Java Web开发技术，在开源代码和开源框架的有力推动下，得到了快速的发展，影响并改变着整个互联网技术生态链条，也深刻地影响着以Android为代表的移动互联网生态群落的当前和未来的发展趋势。

现在，Java Web技术在企业项目开发中的应用越来越广泛，围绕Java衍生的Web核心技术、Java Web开发框架、设计模式等已经成为技术开发研究者的深入研究领域。同时，其相关的一些核心技术也已经成为院校软件开发相关专业学生未来就业和企业Java开发人员快速提升的必备技术，也被许多开发人员当作一项专项技能来学习和掌握。因而，了解Java Web应用后面的核心技术、技术原理及应用对很多人而言非常重要。

诚然，目前市面涉及Java Web核心技术的书籍门类繁多，但通常都为厚重的实践项目案例集成，非常繁琐且没有把理论分析和实践技术进行结合，更没有对整个Java Web开发涉及的核心技术全面、整体、由浅入深进行介绍的书籍。另外，由于本人身处教育行业，从近年来软件工程技术领域研究和教育研究来看，关于Java Web核心技术的介绍比较通俗易懂，适合没有技术背景的人员阅读。另外，比较全面的Java Web开发方面的书籍较少，能够应用于专业教学、符合专业人才培养、实践能力培养的更少，如何有效地解决这些问题，也是本书撰写的基本动因。

随着Java Web及其应用的不断丰富、充实，Java Web技术涵盖的范围也不再局限于传统的Java技术和人们常说的Web技术，它是一个不断扩展的技术范畴，一直以来，本人想就其与Java Web开发框架和Java Web技术的前生今世、渊源及发展现状，做一个统领性的解析和梳理，以便读者和学生对此有个初步的认识和了解。但由于个人认知的局限和Java Web技术门类的琐细、繁多，本书只对Java Web核心的一些关键技术及应用做了详述，其涵盖了软件架构模式、开发模式、设计模式和应用服务器等顶层设计内容，这部分内容通常被大多数人用来在设计模式的书籍中进行讲解，并没有结合具体应用来进行分析详述。实践中，通常大多数人在学习设计模式后，在具体工程、案例中并不能很好地应用，而另一方面，初级入门的代码开发者，关于设计模式在工程中如何贯彻应用，并不能有一个清晰的思路。无论哪一种情况，最后结果都是不能对它们实现有效的结合，这也是本书为什么把设计模式、架构模式设计加入Java Web核心技术的初衷，希望能帮助初学者在走进、学习和使用Java Web技术来盖大楼之前，知道如何建设大楼的框架，而不要拘泥于某一细节技术，而放弃全局的视野，迷失在为学技

术而学技术的浩瀚的技术之海，无法自拔。

基于此，本书的 Java Web 核心技术包含了初级、中级和高级三方面的核心技术。初级技术包含了页面设计、创建的初级技术和 JavaScript 技术；中级技术包含了 JSP 技术、JDBC 常用及高级技术和 XML 技术；高级技术包含了 Struts 初级技术、Struts 高级技术、Hibernetate 技术、Spring 初级技术、Spring 高级技术以及 Spring 和 Struts 的结合集成 Java Web 项目开发技术框架 SSH(Struts-Spring-Hibernate)。其目的是为了使初学者和读者对整个 Java Web 技术从设计模式到初、中、高级技术有个了解和认识性的循序渐进的学习，学习者在阅读中会发现，技术的讲解是一个方面，更为重要的是符合人们认知规律的螺旋式渐进技术体系安排，有利于读者培养理论和技术应用有效结合的学习模式，通过项目技术引导可以使读者明白为什么而学技术(学习的目标性)，技术核心要点、原理之间的关系及衔接(学哪些内容及学习内容之间先后次序关系)，同时，也可以通过技术应用了解自己的学习深入程度及效果(学的效果如何)。

此外，由于移动互联网方面应用的快速发展，Java Web 核心技术的一个分支在移动互联网领域也得到了快速的发展，但限于篇幅，本书并没有对此进行介绍，本书只是在其基础、中级、高级技术的基础上，对 Java Web 核心技术中的一个具有代表性的开源框架系统 SSH 及其在项目中的应用，进行了深入的技术解析和详解，此方面包含了 4 个部分的内容。

第一部分包含了 Struts-Spring-Hibernate 框架概述、Struts 基础、Struts 2 标签、Struts 设计模式、工作流程、表达式、Struts 高级技术、Struts 拦截器和文件上传等。

第二部分包含了 Hibernate 基础、Hibernate 实体关系映射、Hibernate 查询语言和表的设计等。

第三部分包含了 Spring 技术、IoC 模式、Bean 的应用和 Spring Bean 的开发技术等。

第四部分包含了 Spring 持久层、Spring AOP、面向方面编程和事务处理、Struts-Spring 框架技术集成、Struts-Spring-Hibernate 框架集成等。

总而言之，本书内容技术体系及应用初步实现了自己的初衷，是对整个 Java Web 技术体系过往的深入导引，但由于自己水平和认识的局限，难免有遗漏和错误之处，希望阅读者能对此进行反馈，共同推动 Java Web 技术在国内的推广和应用，促进开源技术的快速发展。

致谢

感谢学院领导一直以来的支持和帮助，感谢创新团队项目对我的支持和帮助，是你们促使了这项工作的进展和人才培养教育的落地。

特别感谢我的家人，书稿的写作是一项耗时耗力的工作，没有你们的支持，这项工作基本无法完成，你们的鼓励促使我不断前行和进步。

感谢阅读本书的读者，您的建议和反馈，将是本书完善的基础，希望与你们一起促进国内开源技术的普及和推广。

毋建军

2015 年 1 月

目　录

第 1 章　Java Web 技术概述 …… 1

1.1　Java Web 开发模式 …… 1
1.1.1　软件架构模式(C/S、B/S) …… 1
1.1.2　软件初期设计模式 …… 3
1.1.3　MVC 模式 …… 5
1.1.4　框架模式与设计模式 …… 6
1.2　Java Web 应用服务器 …… 7
1.2.1　Apache 服务器 …… 7
1.2.2　Tomcat 服务器 …… 8
1.2.3　WebSphere 服务器 …… 8
1.2.4　WebLogic 服务器 …… 8
1.2.5　Resin 服务器和 JBoss 服务器 …… 9
1.3　Java Web 服务器安装、测试 …… 9
1.3.1　Apache 服务器 …… 9
1.3.2　Tomcat 服务器 …… 17
1.3.3　WebSphere 服务器 …… 21
1.4　Java Web 开发环境搭建 …… 22
1.4.1　开发工具与环境 …… 22
1.4.2　开发工具集成 …… 23
1.4.3　创建部署 Web 程序 …… 26
1.5　小结 …… 28

第 2 章　Java Web 基础 …… 29

2.1　HTML 语言 …… 29
2.1.1　HTML 简介 …… 29
2.1.2　HTML 基本结构 …… 29
2.1.3　HTML 常用标签 …… 30
2.2　JavaScript 技术 …… 32
2.2.1　JavaScript 简介 …… 32
2.2.2　JavaScript 表单应用 …… 33
2.2.3　JavaScript 正则表达式 …… 35

2.3 Servlet 技术 …… 39
2.4 JSP 技术 …… 43
2.4.1 JSP 技术简介 …… 43
2.4.2 JSP 页面元素 …… 44
2.4.3 JSP 内置对象 …… 47
2.4.4 JSP 异常处理 …… 51
2.5 小结 …… 54

第 3 章 JDBC 技术 …… 55

3.1 JDBC 技术简介 …… 55
3.1.1 JDBC 简介 …… 55
3.1.2 JDBC API …… 56
3.2 JDBC 驱动和数据库访问 …… 57
3.2.1 JDBC 驱动 …… 57
3.2.2 JDBC 访问数据库 …… 57
3.3 JDBC 数据库高级应用 …… 62
3.3.1 JDBC SQL 异常处理 …… 62
3.3.2 事务处理 …… 65
3.3.3 元数据 …… 67
3.3.4 数据源应用 …… 68
3.4 小结 …… 74

第 4 章 XML 技术 …… 75

4.1 XML 技术简介 …… 75
4.1.1 XML 简介 …… 75
4.1.2 XML 特性 …… 76
4.2 XML 组成、规范 …… 77
4.2.1 XML 文档结构 …… 77
4.2.2 XML 基本语法 …… 78
4.2.3 XML 标记 …… 79
4.2.4 XML 元素和属性 …… 80
4.2.5 XML DTD 格式 …… 81
4.2.6 XML Schema 格式 …… 83
4.3 XML 技术应用 …… 85
4.3.1 XML DTD 应用 …… 85
4.3.2 XML Schema 应用 …… 86
4.4 XML 解析 …… 88
4.4.1 DOM 解析 …… 88
4.4.2 SAX 解析 …… 94
4.4.3 DOM4J 解析 …… 99

4.5 小结 …… 103

第5章 Struts技术 …… 104

5.1 Struts基础 …… 104
5.1.1 Struts技术简介 …… 104
5.1.2 Struts模型映射 …… 104
5.2 Struts 2框架及工作流程 …… 106
5.2.1 Struts 2框架 …… 106
5.2.2 Struts 2的工作流程 …… 106
5.2.3 Struts 2基本配置及简单应用 …… 107
5.2.4 Struts 2常用配置 …… 111
5.3 创建Controller组件 …… 125
5.3.1 FilterDispatcher …… 125
5.3.2 Action的开发 …… 126
5.3.3 Model驱动 …… 129
5.4 Model组件创建 …… 129
5.5 View组件创建 …… 130
5.6 小结 …… 131

第6章 Struts 2标签 …… 132

6.1 Struts 2标签简介 …… 132
6.2 一般标签(非UI标签) …… 133
6.2.1 控制标签 …… 133
6.2.2 数据输出标签 …… 135
6.3 UI标签 …… 139
6.3.1 表单标签 …… 139
6.3.2 非表单标签 …… 139
6.3.3 综合应用 …… 141
6.4 EL表达式语言 …… 143
6.4.1 EL基本用法 …… 144
6.4.2 OGNL表达式 …… 144
6.5 小结 …… 150

第7章 Struts高级技术 …… 151

7.1 Struts 2国际化 …… 151
7.1.1 Struts 2国际化方式 …… 151
7.1.2 参数化国际化字符串 …… 156
7.1.3 Struts 2定位资源属性文件顺序 …… 159
7.1.4 其他加载国际化资源文件的方式 …… 161
7.1.5 国际化应用实例 …… 165

7.1.6 数据库中文问题的处理 …… 166
7.2 Struts 2 下快捷地选择或切换语言 …… 168
7.3 Struts 2 类型转换 …… 169
7.4 数据验证 …… 171
7.4.1 使用 Action 的 validate()方法 …… 172
7.4.2 使用 Validation 框架验证数据 …… 174
7.5 Struts 2 拦截器 …… 179
7.5.1 Struts 2 拦截器概述 …… 179
7.5.2 拦截器的应用 …… 180
7.6 Struts 2 文件传输 …… 184
7.6.1 创建上传、下载页面 …… 184
7.6.2 创建文件上传、下载 Action 处理类 …… 186
7.6.3 配置 struts.xml 文件 …… 189
7.6.4 错误信息输出 …… 191
7.7 小结 …… 191

第 8 章 Hibernate 技术 …… 192

8.1 Hibernate 概述 …… 192
8.2 Hibernate 对象/关系数据库映射(单表) …… 194
8.2.1 持久化层 …… 194
8.2.2 Session 操作方法 …… 210
8.3 Hibernate 实体关系映射(多表) …… 211
8.3.1 一对一关系 …… 211
8.3.2 一对多、多对一关系 …… 215
8.3.3 多对多关系 …… 218
8.4 Hibernate 继承策略 …… 222
8.5 Hibernate 应用开发 …… 227
8.6 小结 …… 230

第 9 章 Spring 技术 …… 231

9.1 Spring 概述 …… 231
9.2 IoC(控制反转)模式 …… 233
9.3 Spring 核心容器 …… 235
9.3.1 BeanFactory …… 235
9.3.2 BeanWrapper …… 237
9.3.3 ApplicationContext …… 238
9.3.4 Web Context 应用 …… 241
9.4 Bean 应用 …… 242
9.4.1 Bean 定义及应用 …… 242
9.4.2 Bean 的生命周期 …… 246

9.4.3 Bean 的依赖方式 …… 250
9.4.4 集合注入的方式 …… 251
9.5 Spring Bean 应用开发 …… 253
9.6 小结 …… 256

第 10 章 Spring 高级技术与集成 …… 257

10.1 Spring 持久层 …… 257
10.1.1 数据源的注入 …… 258
10.1.2 Spring 定时器 …… 262
10.2 Spring AOP …… 264
10.2.1 AOP 概念和通知 …… 264
10.2.2 Spring 切入点 …… 269
10.2.3 AOP 基本应用 …… 270
10.3 创建 AOP 代理 …… 274
10.4 Spring 事务处理 …… 276
10.4.1 编程式事务处理 …… 277
10.4.2 声明式事务处理 …… 279
10.5 Spring 和 Struts 集成应用 …… 282
10.6 Struts-Spring-Hibernate 的集成应用 …… 287
10.7 小结 …… 299

参 考 文 献 …… 300

第 1 章　Java Web 技术概述

随着技术的快速发展和广泛应用，Java Web 技术已经成为当前 Web 开发领域应用最为广泛和流行的技术，广泛地应用于各种电子商务网站、政府电子政务及其他软件的开发，如京东、当当、Apache 平台下的开源的项目 Nutch 等。Java Web 技术是以 Java 为核心基础的 Web 应用开发技术的统称，其涵盖了客户端开发技术（HTML 及 CSS 样式、JSP 语言、XML 技术、JavaScript 技术、Servlet、HTTP 协议、JQuery 技术、JavaFX、Ajax 技术（Ajax 框架 Prototype、JQuery））、服务器端开发技术（Java 核心技术、UML 建模开发、JDBC、Java Web 开发框架（SSH、RIA））、数据库开发技术和 SQL 语言、Java Web 项目管理技术及工具（ANT、CVS、SVN）、Java Web 测试技术及工具（JMeter、Junit、Log4j、LoadRunner）等。因其涉及的技术非常广泛，本书只对其中 Java Web 部分核心技术进行深入的解析和应用，使其对 Web 开发有兴趣的程序员和学习者有深入的帮助和了解。

本章将就 Java Web 开发的基础、相关的基本概念、关键技术的当前进展、开发环境的搭建、Web 开发中常用的设计模式（C/S、B/S 模式）和 Java Web 开发技术架构进行阐述，并就其运行的原理及其技术模式进行比较和分析。

1.1　Java Web 开发模式

1.1.1　软件架构模式（C/S、B/S）

随着全球网络开发、互联、信息共享技术要求的不断提高，以前软件开发的 C/S（客户服务端）模式（单机服务软件），已经远远不能满足人们的需要和信息共享发展的要求，其中，以 B/S（浏览器/服务器）模式为代表的新型应用模式被广泛应用。在此模式下用户可以通过 WWW 浏览器去访问 Internet 上的文本、数据、图像、动画、视频点播和声音信息，这些信息都是由许许多多的 Web 服务器产生的，而每一个 Web 服务器又可以通过各种方式与数据库服务器连接，大量的数据实际存放在数据库服务器中。客户端除了 WWW 创览器，一般无须任何用户程序，只需从 Web 服务器上下载程序到本地来执行，在下载过程中若遇到与数据库有关的指令，由 Web 服务器交给数据库服务器来解释执行，并返回给 Web 服务器，Web 服务器又返回给用户。

C/S 模式主要由客户应用程序（Client）、服务器管理程序（Server）和中间件（Middleware）3 个部件组成。客户端应用程序是 Web 系统中被用户用来与数据进行交互的前端。服务器程序负责有效地管理系统资源，其主要工作是当多个客户并发地请求服务器上的相同资源时，对这些资源进行最优化管理。中间件负责联结客户端应用程序与服务器管理程序，协同完成

一个作业，以满足用户查询管理数据的需求。

浏览器/服务器(Browser/Server，B/S)模式是一种以 Web 技术为基础的新型的系统平台模式。把传统 C/S 模式中的服务器部分分解为一个数据服务器与一个或多个应用服务器(Web 服务器)，从而构成一个三层结构的客户服务器体系。

第一层客户端是用户与整个系统的接口，客户的应用程序简化为一个通用的浏览器软件，如 IE、Chrome、火狐等，浏览器将 HTML 代码转化成图文并茂的网页。网页具备一定的交互功能，允许用户在网页提供的申请表上输入信息提交给后台第二层的 Web 服务器，并提出处理请求。

第二层 Web 服务器将启动相应的进程来响应这一请求，并动态生成一串 HTML 代码，其中嵌入处理的结果，返回给客户端的浏览器。如果客户端提交的请求包括数据的存取，Web 服务器还需与数据库服务器协同完成这一处理工作。

第三层数据库服务器的任务类似于 C/S 模式，负责协调不同的 Web 服务器发出的 SQ 请求，管理数据库。

在实践应用软件开发中，应用软件开发的层次，通常分为 3 层，分别是表现层(P)、逻辑层(B)和数据层(D)，它们各自的作用主要表现为：

- 表现层——向访问客户端展现 UI；
- 逻辑层——完成客户需求的程序功能；
- 数据层——保存业务数据的持久化。

无论基于 CS/BS，都需要有与客户产生交互作用的层面，因而，C/S 模式开发通常为用来访问服务程序的客户端程序，需要独立开发，则称为 CS 模式的开发；如利用已有的 Browser，并在此基础上开发 Browser 可以运行的程序，称为 B/S 模式。基于 B/S 结构的软件开发层次通常为：

- P：浏览器→HTML
- B：应用程序服务器→ASP、ASP. NET、JSP、Servlet、Python、CGI、…
- D：关系型数据库→Table

在此层次中，HTML 通常只负责显示用户需要呈现的数据或者收集客户提交的数据信息。

(1) B/S 模式的优势主要有以下几方面。

① 简化了客户端，它无须像 C/S 模式那样在不同的客户机上安装不同的客户应用程序，而只需安装通用的浏览器软件，这样不但可以节省客户机的硬盘空间与内存，而且使安装过程更加简便、网络结构更加灵活。

② 简化了系统的开发和维护，系统的开发者无须再为不同级别的用户设计开发不同的客户应用程序，只需把所有的功能都实现在 Web 服务器上，并就不同的功能为各个组别的用户设置权限就可以了。各个用户通过 HTTP 请求在权限范围内调用 Web 服务器上不同处理程序，从而完成对数据的查询或修改。相对于 C/S，B/S 的维护具有更大的灵活性，当其变化时，它无须再为每一个现有的客户应用程序升级，而只需对 Web 服务器上的服务处理程序进行修订。

③ 使用户的操作变得更简单，对于 C/S 模式，客户端应用程序有自己特定的使用流程，使用者需要接受专门培训。而采用 B/S 模式时，客户端只是一个简单易用的浏览器软件，无论是决策层还是操作层的人员都无须培训，就可以直接使用。B/S 模式的这种特性，还使 MIS

系统维护的限制因素更少。B/S 特别适用于网上信息发布，这是 C/S 所无法实现的。

(2) C/S 模式的优势主要有以下几方面。

① 交互性强是 C/S 自身的优势。在 C/S 中，客户端有一套完整的应用程序，在出错提示、在线帮助等方面都有强大的功能，并且可以在子程序间自由切换。B/S 虽然由 JavaScript、VBScript 提供了一定的交互能力，但与 C/S 的一整套客户应用相比太有限了。

② C/S 模式提供了更安全的存取模式。由于 C/S 是配对的点对点的结构模式，采用适用于局域网、安全性比较好的网络协议，安全性可以得到较好的保证。而 B/S 采用点对多点、多点对多点这种开放的结构模式，并采用 TCP/IP 这一类运用于 Internet 的开放性协议，其安全性只能靠数据服务器上管理密码的数据库来保证。

③ 采用 C/S 模式将降低网络通信量。B/S 采用了逻辑上的 3 层结构，而在物理上的网络结构仍然是原来的以太网或环形网，这样，第一层与第二层结构之间的通信、第二层与第三层结构之间的通信都需占用同一条网络线路。而 C/S 只有两层结构，网络通信量只包括 Client 与 Server 之间的通信量。所以，C/S 处理大量信息的能力是 B/S 所无法比拟的。

④ 此外，由于 C/S 在逻辑结构上比 B/S 少一层，对于相同的任务，C/S 完成的速度总比 B/S 快，使得 C/S 更利于处理大量数据。

另外，在实践开发中，有些需要采用 C/S 模式与 B/S 模式相结合的方案，开发者根据一定的原则，将系统的所有子功能分类，决定哪些子功能适合采用 C/S，哪些适合采用 B/S，针对不同的模式进行开发、部署。在软件维护阶段，针对不同模式的子功能应采取不同维护方式。

1.1.2　软件初期设计模式

在早期的 Web 开发中，因为业务比较简单，软件开发中开发者通常将大部分 Web 应用程序用像 ASP、PHP 等过程化语言来创建。它们将像数据库查询语句这样的数据层代码和像 HTML 这样的表示层代码混在一起，并没有上述软件开发层次 3 层的划分。用户数据的呈现及输入的接收、封装、验证、处理，以及对数据库的操作，都放在 JSP 页面中。此时的 Web 软件开发代码混杂在一起，即双层设计模式：Browser → JSP(Web Server，DataBase Server)，JSP 端充当 Web 服务器和数据库服务器的角色。

随着业务越来越复杂，人们开始考虑更好地利用 OOP 来解决上述存在的混杂问题。实践中有人发现把业务逻辑抽取出来，并形成与显示和持久化无关的一层，能够让业务逻辑清晰，产品更便于维护，即 SUN 开始倡导并推行的 JSP 初期模型开发模式：多层设计模式。

多层设计模式，具体可分为以下几种：

Browser→JSP(Web Server)→DataBase Server

Browser→JSP(Web Server)→beans→DataBase Server

Browser→Servlet→JSP→beans→DataBase Server

下面就它们进行详细介绍。

1. 传统的 JSP 初期模型(JSP Model 1)

传统的 JSP 初期开发模型中，JSP 是独立的，自主完成所有的任务，其模型如图 1-1 所示。

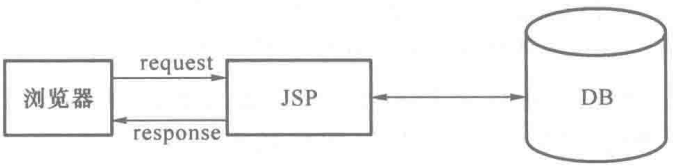

图 1-1　JSP 初期模型

JSP 初期模型的开发方式中，并没有对数据如何持久化给出建议。如图 1-1 所示，在许多产品开发中，产品是以数据库为中心进行架构和设计的。在其产品设计中，虽然也有 DAO 层，但是职责不清，其对 DAO 层的职责简单定位为增删改查。尤其随着业务逻辑变得越来越复杂，复杂的对象关系很难理清，如何把数据存储起来（通常的情况下是存到关系型数据库中）是一个棘手的问题。在此过程中，软件设计中引入面向对象设计的思想，即数据持久化。即在自己的应用中添加一个新的层，持久化层来专门负责对象状态的持久化保存及同步。持久化意味着对关系型数据库的依赖减少。通常，有经验的软件开发者会将数据应用从表示层分离出来，后续讲到的设计模式 MVC 从根本上强制性的将其分开。在此过程中，有人提出了改进后的 JSP 模型——JSP Model 改进模型。

2. 改进的 JSP Model 初期模型（JSP Model 1 改进）

针对 JSP 初期模型数据应用和代码混杂的缺陷，有人提出了 JSP 页面与 JavaBean 协作响应浏览器页面请求，并 response 回复完成任务的 JSP Model 改进模型，其工作模式如图 1-2 所示。

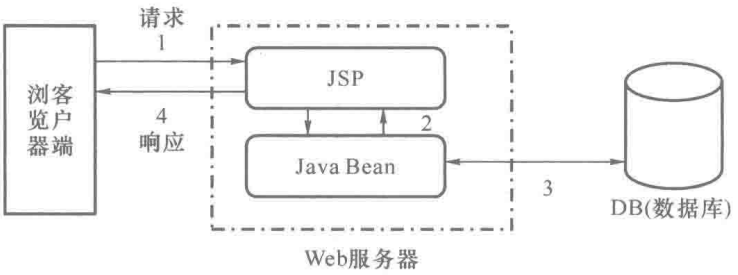

图 1-2　JSP Model 1 改进模型工作模式

从上述的 JSP Model 初期模型可以看出，其实现比较简单，适用于快速开发小规模项目。但就项目产品工程化来分析，其缺陷显而易见，JSP 页面充当了 View 和 Controller 两种角色，将控制逻辑和表现逻辑混杂在一起，从而导致其代码的重用性较低，对应用的后期扩展和维护都增加了难度。而改进后的 JSP Model 模型，增加了 JavaBean，把数据存储及数据库打交道的功能，分解给 JavaBean 进行完成。

早期有大量采集 JSP 技术开发出来的 Web 应用，都采用了 JSP Model 初期架构。

3. JSP Model 2

随着业务变得越来越复杂，在 JSP Model 1 改进的基础上，针对上述的缺陷，出现了 JSP Model 2 架构，JSP Model 2 在 JSP、JavaBeans 的基础上，增加了 Servlet 角色，在此结构中：JSP 负责生成动态网页，只用作显示页面；Servlet 负责流程控制，用来处理各种请求的响应及分派；JavaBeans 负责业务逻辑，实现对数据库的各种操作。

JSP Model 2 的交互过程：用户通过浏览器向 Web 应用中的 Servlet 发送请求，Servlet 接收到请求后实例化 JavaBeans 对象，调用 JavaBeans 对象的方法，JavaBeans 对象返回从数据库中读取的数据，Servlet 选择合适 JSP，并且把从数据库中读取的数据通过这个 JSP 进行显示，最后 JSP 页面把最终的结果返回给浏览器。其交互过程工作模式如图 1-3 所示。

JSPModel 2 已经是 MVC 设计思想下的架构，由此引入了 MVC 模式，使 JSP Model 2 具有组件化的特点，更适用于大规模应用的开发，但也增加了应用开发的复杂程度。从 JSP Model 2 模型理论上看，其已经具备了 Web 三层架构的模式。但在实际开发产品中，我们发现在上述的 UI 层和业务层之间有交叉逻辑。这些交叉逻辑即不属于业务逻辑层管理的范畴，也不属于 UI 层管理的范畴，如页面跳转、表单数据的验证及封装、页面的国际化（在 JSP 页面根据用户的配置或请求信息判断应该为该用户提供哪一种语言的页面信息）等，这些交叉

逻辑如何分配，交给业务逻辑层还是 UI 层，如何实现业务逻辑层和 UI 层的分离呢？下述的 MVC 模式便为解决上述模式中存在的问题，而得到广泛的应用。

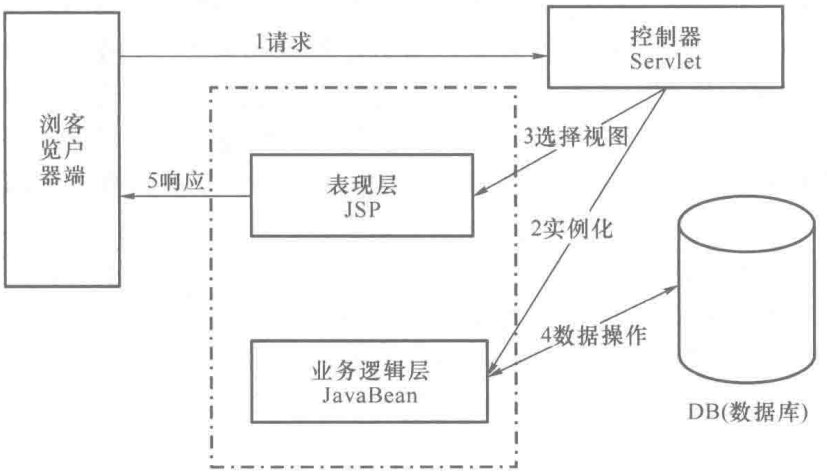

图 1-3　JSP Model 2 交互工作模式

1.1.3　MVC 模式

MVC 模式(Model-View-Controller)，即“模型-视图-控制器”。MVC 是 Xerox PARC 在 20 世纪 80 年代为编程语言 Smalltalk-80 发明的一种软件设计模式，后来被推荐为 Oracle 下 Sun 公司 Java EE 平台的设计模式。它开始是存在于桌面程序中的，M 是指业务模型，V 是指用户界面，C 则是控制器，使用 MVC 的目的是将 M 和 V 的实现代码分离，从而使同一个程序可以使用不同的表现形式，如用户表格数据的提交，统计数据可以分别用柱状图、饼图等表示。C 存在的目的则是确保 M 和 V 的同步，一旦 M 改变，V 应该同步更新。MVC 是一个框架模式，它强制性地使应用程序的输入、处理和输出分开。使 MVC 应用程序被分成 3 个核心部件：模型、视图和控制器。它们各自处理自己的任务。最典型的 MVC 就是上述的 JSP+Servlet+JavaBean 的模式。下面就其核心组件的作用和功能进行详细解析。

1. 视图

视图(View)是用户看到并与之交互的界面。对老式的 Web 应用程序而言，视图就是由 HTML 元素组成的界面，在新式的 Web 应用程序中，HTML 依旧在视图中扮演着重要的角色，但如 Adobe Flash、XHTML、XML/XSL、WML 等一些标识语言及 Web Services 等新技术已经广泛应用。一个 Web 应用可能有很多不同的视图，MVC 设计模式对于视图的处理仅限于视图上数据的采集和处理，以及用户的请求，但不包括在视图上的业务流程的处理。业务流程的处理交给模型(Model)处理，如一个订单的视图只接受来自模型的数据并显示给用户，以及将用户界面的输入数据和请求传递给控制和模型。

2. 模型

模型即业务流程/状态的处理以及业务规则的制定。业务流程的处理过程对其他层而言是黑箱操作，模型接受视图请求的数据，并返回最终的处理结果。业务模型的设计是 MVC 最主要的核心。以前流行的 EJB 模型就是一个典型的应用例子，模型拥有最多的处理任务，被模型返回的数据是中立的，就是说模型与数据格式无关，这样一个模型能为多个视图提供数据。此外，业务模型还有一个很重要的模型是数据模型。数据模型主要指实体对象的数据保存(持续化)，例如，将一张订单保存到数据库，从数据库获取订单，即可以将此模型单独列出，

所有有关数据库的操作都限制在该模型中。

3. 控制器

控制器接收用户的输入并调用模型和视图去完成用户的需求，所以当单击 Web 页面中的超链接和发送 HTML 表单时，控制器本身不输出任何东西和作任何处理。它只是接收请求并决定调用哪个模型构件去处理请求，然后再确定用哪个视图来显示返回的数据。控制器起着一个分发器的作用，所以当用户点击一个链接，控制层接受请求后，并不处理业务信息，它只把用户的信息传递给模型，告诉模型做什么，选择符合要求的视图返回给用户。因此，一个模型可能对应多个视图，一个视图可能对应多个模型。

模型、视图与控制器的分离，使得一个模型可以具有多个显示视图。如果用户通过某个视图的控制器改变了模型的数据，所有其他依赖于这些数据的视图都会发生变化。因此，无论何时发生了何种数据变化，控制器都会将变化通知所有的视图，导致显示的更新，实际上是一种模型的变化——传播机制。模型、视图、控制器及数据库之间的交互关系，如图 1-4 所示。视图中用户的输入被控制器解析后，控制器改变状态激活模型，模型根据业务逻辑维护数据，从数据库提取数据或写入数据，并通知视图，其对应的数据已经发生变化，视图得到消息通知后，从模型中获取新的数据并进行更新显示。

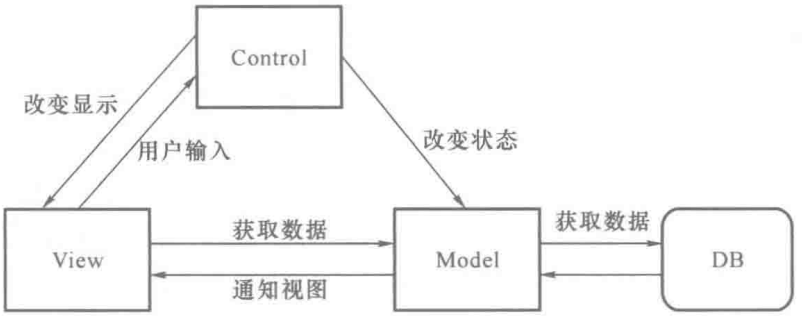

图 1-4　模型、视图、控制器及数据库的交互关系

如上所述 MVC 模式的关键是实现了视图和模型的分离。其实现原理为：MVC 模式通过建立一个“发布-订阅”(Publish-Subscribe)的机制来分离视图和模型。发布-订阅机制的目标是发布者，其发出通知时并不知道，也不需知道谁是它的观察者，可以有任意数目的观察者订阅消息并接收通知。其优点主要有：

- 多视图表示，一个模型提供不同的多个视图表现形式，也能够为一个模型创建新的视图而无须重写模型，一旦模型的数据发生变化，模型将通知有关的视图，每个视图相应地刷新自己；
- 模型可复用，因为模型是独立于视图的，所以可以把一个模型独立地移植到新的平台工作；
- 提高开发效率，在开发界面显示部分时，仅需要考虑的是如何布局一个好的用户界面，开发模型时，只需考虑业务逻辑和数据维护，从而能使开发者专注于某一方面的开发，提高开发效率。

MVC 模式中关键点利用了观察者模式(Observer)，而观察者模式实现了发布-订阅(Publish-Subscribe)机制，并能完成视图和模型的分离。其设计模式关系还涉及了组合模式(Composite)、策略模式(Strategy)，关于这 3 种模式由于篇幅的关系，本书不再详细介绍。

1.1.4　框架模式与设计模式

MVC 是一种框架模式，而不是一种设计模式。框架模式与设计模式，实际上完全是不同的概

念。框架通常是指代码重用，而设计模式是指设计重用，架构则介于两者之间，部分代码重用，部分设计重用，有时分析也可重用。在软件生产中有 3 种级别的重用：内部重用，即在同一应用中能公共使用的抽象块；代码重用，即将通用模块组合成库或工具集，以便在多个应用和领域都能使用；应用框架的重用，即为专用领域提供通用的或现成的基础结构，以获得最高级别的重用性。

框架模式与设计模式虽然相似，但却有着根本的不同。设计模式是对在某种环境中反复出现的问题以及解决该问题的方案的描述，它比框架更抽象；框架可以用代码表示，也能直接执行或复用，而对模式而言只有实例才能用代码表示；设计模式是比框架更小的元素，一个框架中往往含有一个或多个设计模式，框架总是针对某一特定应用领域，但同一模式却可适用于各种应用。可以说，框架是软件，而设计模式是软件的知识。

常见的框架模式有：MVC、MTV、ORM、MVP 等。框架有：基于 PHP 语言的 smarty 框架，基于 C++语言的 QT、MFC、gtk 等，基于 Java 语言的 SSH、SSI 等。本书后续将对 SSH 框架进行深入的解析。

常用的设计模式有：工厂模式、适配器模式、策略模式、观察者模式等。

1.2　Java Web 应用服务器

JavaWeb 服务器是运行及发布 Java Web 应用的容器，只有将开发的 Web 项目放置到该容器中，才能使网络中的所有用户通过浏览器进行访问。开发 Java Web 应用所采用的服务器主要是与 JSP/Servlet 兼容的 Web 服务器，比较常用的 Java Web 服务器有 Apache、Tomcat、Resin、JBoss、WebSphere 和 WebLogic 等，下面将分别进行介绍。

1.2.1　Apache 服务器

Apache HTTP Server(Apache)是 Apache 软件基金会的一个开放源码的网页服务器，可以在大多数计算机操作系统中运行，由于其多平台和安全性被广泛使用，是当前最流行的 Web 服务器端软件之一。它快速、可靠并且可通过简单的 API 扩展，将 Java Web 及 Perl/Python 等解释器编译到服务器中。目前的版本是 2014 年 7 月发布的 Apache 2.4.10，其下载网址为：http://httpd.apache.org/，如图 1-5 所示。

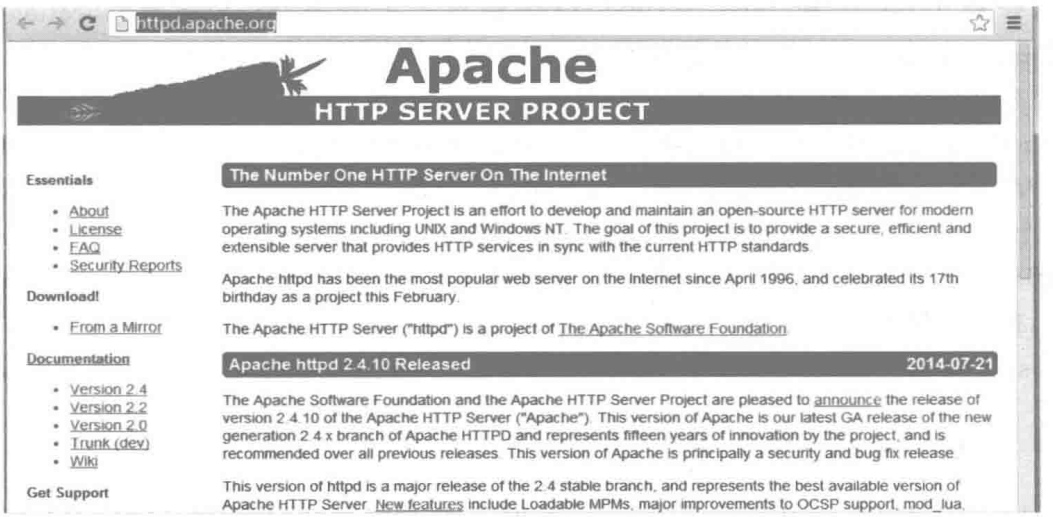

图 1-5　Apache 服务器下载网站