

精通

Struts 2

基于MVC的Java Web 应用开发实战

陈云芳 编著

核心主题：Struts 2、Spring、Hibernate、Ajax

内容覆盖了Struts 2基础、Struts 2框架核心和Struts 2高级特性，流行框架整合开发等知识

三大综合案例开发演示，帮助读者提高项目开发水平

网络商店系统（Struts 2+Hibernate+Velocity）

网络文件管理系统（Struts 2+Hibernate+FreeMarker）

网上购书系统（Struts 2+Hibernate+Ajax）

 人民邮电出版社
POSTS & TELECOM PRESS



CD-ROM

精通

TP312/2871D

2008

Struts 2

基于MVC的Java Web 应用开发实战

陈云芳 编著

人民邮电出版社
北京

图书在版编目 (CIP) 数据

精通 Struts 2: 基于 MVC 的 Java Web 应用开发实战 / 陈云芳
编著. —北京: 人民邮电出版社, 2008.7
ISBN 978-7-115-17877-0

I. 精… II. 陈… III. ①软件工具—程序设计②JAVA
语言—程序设计 IV. TP311.56 TP312

中国版本图书馆 CIP 数据核字 (2008) 第 042569 号

内 容 提 要

本书由浅入深、循序渐进地介绍了使用 Struts 2 框架进行 Web 应用开发的基本原理和方法。

全书共分为 4 篇, 第一篇 Struts 2 快速入门, 主要介绍了基本的 Web 开发技术, Struts 2 的基础知识, 并通过简单应用实例, 讲解了 Struts 2 的框架结构。第二篇 Struts 2 框架核心, 包括 Action、Interceptor、Result、多视图和数据持久化支持等方面的知识。第三篇 Struts 2 高级特性, 介绍了数据验证、类型转换和国际化等方面的内容。第四篇 Struts 2 综合项目实践, 介绍了 3 个综合实例 (网络商店系统、网络文件管理系统、网络购书系统) 的开发过程, 每个实例均按照需求分析、项目设计、代码开发、程序测试及项目部署等实际开发流程编写, 力求使读者在学习理论的同时, 能够进一步通过现有的程序实例加深理解, 提升实战能力。

随书附赠光盘上有本书全部已经调试通过的程序源代码。

本书重点突出, 理论与实践并重, 适合于具有一定 Java 基础, 进行 Web 应用开发的程序员阅读, 也可作为相关培训机构的教材, 以及高等院校相关专业师生的参考用书。

精通 Struts 2——基于 MVC 的 Java Web 应用开发实战

- ◆ 编 著 陈云芳
责任编辑 张 涛
 - ◆ 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>
三河市海波印务有限公司印刷
新华书店总店北京发行所经销
 - ◆ 开本: 787×1092 1/16
印张: 32.75
字数: 803 千字
印数: 1—4 000 册
- 2008 年 7 月第 1 版
2008 年 7 月河北第 1 次印刷

ISBN 978-7-115-17877-0/TP

定价: 65.00 元 (附光盘)

读者服务热线: (010)67132692 印装质量热线: (010)67129223
反盗版热线: (010)67171154

前 言

Java 2 Enterprise Edition (J2EE) 技术自从推出以来得到了广泛认可和应用, 随着多年的技术演变和发展, J2EE 技术已经日趋成熟, 成为当今电子商务的最佳解决方案之一。同时基于 J2EE 的 Web 应用开发如火如荼, 网络上也兴起了一大批开源的框架来支持 J2EE 的开发, Struts 2 是其中最优秀的 Web 应用开发框架之一。Struts 2 框架的出现, 极大地简化了 J2EE 的开发, 现在已经成为企业中使用较为广泛的框架。

本书详细介绍了 Struts 2 的体系结构、框架核心概念以及相应的高级特性话题, 阐述了用 Struts 2 构建 Web 应用的基本原理和开发步骤。最后, 通过 3 个综合实例, 重点向读者展示了如何在实战开发中使用 Struts 2。

内容组织

本书分为 4 篇, 共 21 章。每一篇都专注于使用 Struts 2 开发的不同层次。为了方便读者实战, 每个知识点都有对应的开发实例来配合知识的应用。

第一篇 Struts 2 快速入门, 包括 1~2 章, 向读者展示了 Struts 2 框架的一个快照。这一部分注重讲解 Struts 2 的基础知识, 如框架结构等。

第二篇 Struts 2 框架核心, 包括 3~9 章, 向读者展示了 Struts 2 最核心的控制器 (Action) 和拦截器 (Interceptor), 这两个概念是 Struts 2 最基础、最核心的技术, 是理解 Struts 2 架构的关键所在。

第三篇 Struts 2 高级特性, 包括 10~18 章, 介绍 Struts 2 中数据校验、类型转换、国际化、反转控制等高级的话题, 通过这一部分的学习, 读者对 Struts 2 的应用将上升一个层次。

第四篇综合项目实战, 包括 19~21 章, 通过 3 个完整的综合项目应用的开发, 让读者对使用 Struts 2 开发有一个直观的感受, 同时进一步加强对 Struts 2 的理解。

“网络商店系统 (Struts 2+Hibernate+Velocity)” 构建的全过程, 读者将了解到如何使用所学的 Struts 2 知识开发一个完整的网络商店项目。

“网络文件管理系统 (Struts 2+Hibernate+FreeMarker)” 的整个实例编写过程中, 除了学习使用 Struts 2 框架实现整个系统之外, 还学习了如何解决中文乱码问题以及使用 FreeMarker

编写动态页面。

“网上购书系统 (Struts 2+Hibernate+Ajax+FreeMarker)”，在此案例中会使用 Ajax 中的许多技术，并且能体会到这个技术带来的页面效果。

读者对象

本书是为那些厌烦了复杂 Web 应用开发的程序员准备的。

读者应对 Java 及常用的 Java API 有基本的认识，对 HTML、Servlet、JavaScript 及 CSS 具有基本的了解。如果读者具备一些其他的 Web 开发框架的知识，将能更好地体会本书的精髓。

本书的优势

当前市场上已有的介绍 Struts 2 技术的书中，由于诸多原因存在着以下一些问题。

(1) 框架核心思想只做简单介绍，没有深入到源代码中，缺少相关编程思想介绍，也缺少对应模块的应用实例。

(2) 许多实例开发明显不够具体，草草了事，读者看了之后无所适从，既不能体会到软件开发的过程，也没有提高对框架本身的认识。

本书主要有以下几个特色。

(1) 在介绍 Struts 2 框架的核心内容中，坚持以开放的源代码为核心，深入挖掘其框架的核心思想，精心向读者展示 Struts 2 的精妙理念。

(2) 在 Struts 2 高级特性部分，除了展示一些常用业务逻辑的处理之外，还增加了 Web 应用的开发调试、Struts 2 原理探究以及与其他框架的整合。

(3) 在 Struts 2 的实际应用案例开发中，以迭代开发的方式进行逐步描述，既展现了基于 Struts 2 框架的 Web 应用开发过程，也使得读者对软件开发有了更加深入、细致的体会。

本书编者

本书主编陈云芳博士，使用 Java 相关技术开发超过 8 年，具有丰富的 Web 应用系统开发经验，对 J2EE 架构理解透彻。近年来一直关注于开源的一些 Web 项目，对如何使用 Struts、Spring 和 WebWork 等开发框架进行 Web 应用开发有深刻的体会。同时参与本书编写的还有王震编写第 1~6 章，刘冬东编写第 7~11 章，许欣苒编写第 12~17 章，第 18~21 章由陈云芳、李凌晶和陈奇龙完成。另外高晓东、王蕴杰、蔡径舟、冯骥、李佳恒、吴炯明参与了本书的审校，刘达林、井鹏、孙琪、王赞、李维佳参加了代码的调试。

由于时间有限，书中难免存有疏漏之处，敬请读者批评指正。联系邮箱为：zhangtao@ptpress.com.cn。

编者

2008 年 4 月

目 录

第 1 篇 Struts 2 快速入门

| | |
|---------------------------------|--|
| 第 1 章 Web 应用开发概览.....2 | 与配置..... 20 |
| 1.1 Web 应用与 Web 开发框架.....2 | 2.1.2 MyEclipse 的安装和配置..... 23 |
| 1.1.1 Web 应用开发历程.....2 | 2.1.3 Struts 2 的安装..... 26 |
| 1.1.2 为什么需要 Web 框架.....5 | 2.2 创建配置文件..... 28 |
| 1.2 MVC 模式和 Struts.....8 | 2.2.1 配置 web.xml 文件..... 28 |
| 1.2.1 什么是 MVC 模式.....8 | 2.2.2 配置 Struts 2 的 xml 文件..... 29 |
| 1.2.2 MVC 模式的 Web 框架映射.....9 | 2.2.3 配置 Struts 2 的 properties 文件..... 30 |
| 1.2.3 Struts 2 对 MVC 模式的改进...11 | 2.3 第一个 Action..... 30 |
| 1.3 Struts 2 框架概览..... 13 | 2.3.1 创建 helloworld.action..... 31 |
| 1.3.1 Struts 2 框架的演变..... 13 | 2.3.2 在 struts.xml 中配置 Action..... 31 |
| 1.3.2 Struts 2 的特性..... 14 | 2.3.3 创建视图 hello.jsp..... 31 |
| 1.3.3 与其他 Web 框架的比较..... 17 | 2.3.4 测试 Action..... 32 |
| 1.4 小结..... 19 | 2.4 体验 Struts 2 更高级功能..... 34 |
| 第 2 章 第一个 Struts 2 应用.....20 | 2.4.1 处理输入..... 34 |
| 2.1 Struts 2 开发环境配置.....20 | 2.4.2 执行结果..... 36 |
| 2.1.1 JDK1.5.10 和 Tomcat 的安装 | 2.4.3 拦截器..... 36 |
| | 2.5 小结..... 39 |

第 2 篇 Struts 2 框架核心

| | |
|----------------------|---------------------------|
| 第 3 章 Action..... 42 | 3.1.1 返回类型..... 43 |
| 3.1 Action 简介.....42 | 3.1.2 execute()方法..... 44 |
| | 3.2 Action 接口..... 46 |

| | | | | | |
|-------|----------------------------------|----|-------|---------------|-----|
| 3.3 | ActionSupport 基类 | 46 | 5.2 | 常用的 Result | 84 |
| 3.4 | Action 实例 | 48 | 5.2.1 | Dispatcher | 86 |
| 3.4.1 | 实现 Action 接口的实例 | 48 | 5.2.2 | Redirect | 90 |
| 3.4.2 | 实现 ActionSupport 基类的实例 | 51 | 5.2.3 | Chain | 95 |
| 3.5 | 在 Action 中实现基本校验 | 54 | 5.3 | 其他的 Result | 96 |
| 3.5.1 | 使用 Action 的 execute() 方法 | 54 | 5.4 | 小结 | 97 |
| 3.5.2 | 使用 ActionSupport 的 validate() 方法 | 54 | 第 6 章 | 标签 | 99 |
| 3.5.3 | 使用注释实现校验 | 55 | 6.1 | 标签介绍 | 99 |
| 3.6 | ActionContext | 59 | 6.2 | 自定义标签 | 100 |
| 3.7 | 小结 | 60 | 6.2.1 | 标签的实现 | 100 |
| 第 4 章 | Interceptor | 61 | 6.2.2 | 标签的配置 | 101 |
| 4.1 | Interceptor 基础 | 61 | 6.2.3 | 运行 | 103 |
| 4.1.1 | 理解 Interceptor | 61 | 6.3 | Struts 2 标签概览 | 103 |
| 4.1.2 | Interceptor 何时调用 | 62 | 6.3.1 | 标签简介 | 103 |
| 4.1.3 | Interceptor 接口 | 63 | 6.3.2 | 使用标签的准备 | 104 |
| 4.1.4 | Interceptor 相关类 | 64 | 6.4 | 非 UI 标签 | 105 |
| 4.2 | 使用预定义的 Interceptor | 65 | 6.4.1 | 数据标签 | 105 |
| 4.2.1 | 预定义 Interceptor 类 | 65 | 6.4.2 | 控制标签 | 109 |
| 4.2.2 | LoggingInterceptor 示例 | 68 | 6.4.3 | 其他标签 | 112 |
| 4.3 | 使用自定义的 Interceptor | 70 | 6.5 | UI 标签 | 114 |
| 4.3.1 | 自定义 Interceptor 的配置 | 71 | 6.5.1 | 通用属性 | 115 |
| 4.3.2 | 实现自己的 Interceptor | 71 | 6.5.2 | 简单 UI 标签 | 118 |
| 4.4 | Interceptor 应用实例 | 73 | 6.5.3 | 高级 UI 标签 | 120 |
| 4.4.1 | execAndWait 拦截器源码分析 | 73 | 6.6 | 小结 | 124 |
| 4.4.2 | 项目概述 | 74 | 第 7 章 | 表达式语言 OGNL | 125 |
| 4.4.3 | Action 实现文件 | 74 | 7.1 | OGNL 的基本特性 | 125 |
| 4.4.4 | 页面文件 | 75 | 7.1.1 | 常量的声明 | 126 |
| 4.4.5 | 配置文件 | 77 | 7.1.2 | 方法调用 | 127 |
| 4.4.6 | 运行 | 78 | 7.1.3 | 变量引用 | 127 |
| 4.5 | 小结 | 79 | 7.2 | 操作集合 | 128 |
| 第 5 章 | Result | 80 | 7.2.1 | 操作列表 Lists | 128 |
| 5.1 | 自定义 Result | 80 | 7.2.2 | 操作映射 Maps | 128 |
| 5.1.1 | 实现自定义 Result | 81 | 7.2.3 | 操作数组 Array | 128 |
| 5.1.2 | 配置自己的 Result | 82 | 7.2.4 | 集合操作 | 129 |
| 5.1.3 | 运行测试 Result | 83 | 7.3 | OGNL 的高级特性 | 130 |
| | | | 7.3.1 | 类型转换 | 130 |
| | | | 7.3.2 | 与值栈的联系 | 131 |

| | | | |
|---------------------------------|-----|---------------------------|-----|
| 7.3.3 lamuda 表达式 | 132 | 8.3 JasperReports | 148 |
| 7.3.4 Java 和 OGNL 的比较 | 132 | 8.3.1 JasperReports 介绍 | 148 |
| 7.4 在 JSP 中使用 OGNL | 134 | 8.3.2 JasperReports 实例分析 | 150 |
| 7.4.1 项目概述 | 134 | 8.4 小结 | 153 |
| 7.4.2 工程代码编写 | 134 | 第 9 章 数据持久化——使用 Hibernate | 154 |
| 7.4.3 运行 | 137 | 9.1 数据持久化与 Hibernate 基础 | 154 |
| 7.5 小结 | 137 | 9.1.1 数据持久化基础 | 154 |
| 第 8 章 多视图 | 138 | 9.1.2 Hibernate 基础 | 155 |
| 8.1 Velocity | 138 | 9.2 Hibernate 简单应用 | 156 |
| 8.1.1 Velocity 简介 | 138 | 9.3 Hibernate 核心 | 160 |
| 8.1.2 语法 | 139 | 9.3.1 Hibernate 的核心接口 | 160 |
| 8.1.3 Velocity 实例分析 | 140 | 9.3.2 Hibernate 的配置文件 | 162 |
| 8.2 FreeMarker | 143 | 9.3.3 Hibernate 的对象关系映射 | 163 |
| 8.2.1 FreeMarker 简介 | 143 | 9.3.4 Hibernate 的检索方式 | 163 |
| 8.2.2 语法 | 144 | 9.4 Hibernate 实例 | 164 |
| 8.2.3 FreeMarker 实例分析 | 146 | 9.5 小结 | 177 |
| 8.2.4 FreeMarker 和 Velocity 的比较 | 147 | | |

第 3 篇 Struts 2 高级特性

| | | | |
|-------------------------|-----|-------------------------|-----|
| 第 10 章 校验框架 | 180 | 11.2.1 自带转换器的数据类型 | 202 |
| 10.1 使用 Struts 2 自带的校验器 | 180 | 11.2.2 转换器的源码分析 | 203 |
| 10.1.1 配置拦截器 | 180 | 11.3 自定义转换器 | 206 |
| 10.1.2 配置校验器 | 181 | 11.3.1 编写转换器类 | 206 |
| 10.1.3 自带校验器实例 | 184 | 11.3.2 配置 | 207 |
| 10.2 使用自定义校验器 | 191 | 11.4 高级类型转换 | 208 |
| 10.2.1 创建自定义校验器 | 191 | 11.4.1 空值属性处理 | 208 |
| 10.2.2 自定义校验器的注册 | 192 | 11.4.2 类型转换错误处理 | 209 |
| 10.2.3 自定义校验器实例 | 193 | 11.5 小结 | 210 |
| 10.3 小结 | 196 | 第 12 章 国际化 | 211 |
| 第 11 章 类型转换 | 197 | 12.1 国际化基础 | 211 |
| 11.1 为什么需要类型转换 | 197 | 12.2 Struts 2 与国际化 | 212 |
| 11.1.1 没有类型转换的实例 | 197 | 12.2.1 Struts 2 对国际化的支持 | 212 |
| 11.1.2 类型转换带来的好处 | 201 | 12.2.2 资源包相关 | 212 |
| 11.2 Struts 2 自带的转换器 | 202 | 12.2.3 国际化信息的获取 | 214 |
| | | 12.3 Struts 2 国际化实现 | 214 |

| | | | |
|----------------------------------|------------|-------------------------------|------------|
| 12.3.1 初始页面的国际化 | 214 | 15.2 Quartz 作业调度 | 267 |
| 12.3.2 Action 的国际化 | 216 | 15.2.1 Quartz 简介 | 268 |
| 12.3.3 校验的国际化 | 217 | 15.2.2 作业调度简介 | 268 |
| 12.4 国际化实例 | 217 | 15.2.3 Quartz 应用实例 | 269 |
| 12.4.1 项目配置与开发 | 217 | 15.3 Acegi-Security 用户管理 | 271 |
| 12.4.2 结果测试 | 223 | 15.3.1 Acegi-Security 简介 | 271 |
| 12.5 小结 | 225 | 15.3.2 Acegi 认证过程 | 272 |
| 第 13 章 Struts 2 调试 | 226 | 15.3.3 保护 Web 应用程序 | 274 |
| 13.1 MyEclipse 中的 Java 程序调试 | 226 | 15.4 小结 | 275 |
| 13.1.1 调试配置 | 226 | 第 16 章 Ajax 与 Struts 2 | 276 |
| 13.1.2 调试视图 | 227 | 16.1 Ajax 入门 | 276 |
| 13.1.3 控制程序执行 | 229 | 16.1.1 Ajax 发展过程 | 276 |
| 13.2 调试 Struts 2 程序 | 229 | 16.1.2 Ajax 魅力所在 | 277 |
| 13.2.1 调试入门 | 229 | 16.2 Ajax 核心 | 278 |
| 13.2.2 连接 Struts 2 的源代码 | 231 | 16.2.1 使用 XMLHttpRequest 对象 | 278 |
| 13.3 Struts 2 调试应用 | 234 | 16.2.2 发送请求与处理响应 | 282 |
| 13.3.1 实例的概述 | 235 | 16.2.3 实现基本的 Ajax 技术 | 286 |
| 13.3.2 实例工程的源文件 | 235 | 16.3 Struts 2 与 Ajax 框架的整合 | 290 |
| 13.3.3 调试工程 | 240 | 16.3.1 Dojo 框架 | 290 |
| 13.4 小结 | 243 | 16.3.2 DWR 框架 | 292 |
| 第 14 章 Struts 2 原理探究 | 244 | 16.3.3 使用 JSON 插件 | 296 |
| 14.1 控制反转 (IOC) | 244 | 16.4 Struts 2 的 Ajax 标签 | 301 |
| 14.1.1 什么是控制反转 | 244 | 16.5 小结 | 308 |
| 14.1.2 控制反转的设计方式 | 246 | 第 17 章 Struts 2 测试 | 309 |
| 14.1.3 Struts 2 中的控制反转 | 248 | 17.1 软件测试与 JUnit 基础 | 309 |
| 14.2 源码探究 | 251 | 17.1.1 软件测试基础 | 309 |
| 14.2.1 Struts 2 启动 | 251 | 17.1.2 JUnit 基础 | 310 |
| 14.2.2 Action 处理过程 | 256 | 17.2 JUnit 的使用 | 312 |
| 14.2.3 Interceptor 工作机制 | 260 | 17.2.1 JUnit 的安装 | 312 |
| 14.2.4 ActionContext | 261 | 17.2.2 创建测试类 | 313 |
| 14.3 小结 | 262 | 17.2.3 创建被测试类 | 313 |
| 第 15 章 Struts 2 与其他网络框架整合 | 263 | 17.2.4 运行测试 | 314 |
| 15.1 与 SiteMesh 整合 | 263 | 17.3 在 Eclipse 中使用 JUnit 测试 | |
| 15.1.1 SiteMesh 简介 | 263 | Action | 315 |
| 15.1.2 SiteMesh 的安装 | 264 | 17.3.1 创建 Struts 2 项目 | 315 |
| 15.1.3 SiteMesh 应用实例 | 265 | 17.3.2 创建测试类 | 319 |
| | | 17.3.3 运行测试 | 322 |

| | | | |
|--------------------------|-----|-------------------------|-----|
| 17.4 JUnit 高级..... | 323 | 18.1.1 Spring 概述..... | 327 |
| 17.4.1 类级别的 fixture..... | 323 | 18.1.2 Spring 准备..... | 328 |
| 17.4.2 限时测试..... | 324 | 18.1.3 入门实例..... | 328 |
| 17.4.3 异常测试..... | 324 | 18.2 Spring 基础特性..... | 331 |
| 17.4.4 忽略测试..... | 324 | 18.2.1 依赖注入..... | 331 |
| 17.4.5 测试运行器..... | 325 | 18.2.2 Spring 封装机制..... | 333 |
| 17.4.6 测试套件..... | 325 | 18.3 Spring 高级特性..... | 339 |
| 17.4.7 参数化测试..... | 325 | 18.3.1 Spring MVC..... | 339 |
| 17.5 小结..... | 326 | 18.3.2 数据持久层..... | 344 |
| 第 18 章 Spring 整合..... | 327 | 18.4 小结..... | 346 |
| 18.1 Spring 初步..... | 327 | | |

第 4 篇 Struts 2 综合项目实战

| | | | |
|---|-----|---|-----|
| 第 19 章 网络商店实例编程 (Struts 2+Hibernate+Velocity)..... | 348 | 19.4.3 购物车的功能需求和设计 实现..... | 378 |
| 19.1 项目概述..... | 348 | 19.4.4 实现步骤..... | 379 |
| 19.1.1 网络商店概况..... | 348 | 19.5 迭代四后台管理员功能..... | 386 |
| 19.1.2 网络商店的需求分析..... | 348 | 19.5.1 本阶段的迭代目标..... | 386 |
| 19.1.3 网络商店的业务建模..... | 349 | 19.5.2 本阶段页面概述..... | 388 |
| 19.1.4 迭代式开发介绍..... | 349 | 19.5.3 实现步骤..... | 390 |
| 19.1.5 项目功能模块分割..... | 350 | 19.6 小结..... | 398 |
| 19.2 迭代一游客浏览商品..... | 351 | 第 20 章 网络文件管理系统 (Struts 2+Hibernate+FreeMarker)..... | 399 |
| 19.2.1 本阶段迭代的目标..... | 351 | 20.1 项目概述..... | 399 |
| 19.2.2 本阶段页面概述..... | 352 | 20.1.1 网络文件管理系统概况..... | 399 |
| 19.2.3 商品的数据模型和数据库 设计..... | 354 | 20.1.2 网络文件管理系统需求分析..... | 400 |
| 19.2.4 实现步骤..... | 354 | 20.1.3 网络文件管理系统业务建模..... | 401 |
| 19.3 迭代二用户管理..... | 361 | 20.1.4 项目功能模块分割..... | 401 |
| 19.3.1 本阶段迭代的目标..... | 361 | 20.2 迭代一实现上传、下载、删除 文件..... | 402 |
| 19.3.2 本阶段页面概述..... | 362 | 20.2.1 本阶段迭代的目标..... | 402 |
| 19.3.3 用户数据模型和数据库设计..... | 364 | 20.2.2 本阶段页面概述..... | 403 |
| 19.3.4 实现步骤..... | 364 | 20.2.3 文件的数据模型..... | 406 |
| 19.4 迭代三购物车管理..... | 373 | 20.2.4 实现步骤..... | 406 |
| 19.4.1 本阶段的迭代目标..... | 373 | 20.3 迭代二添加用户管理..... | 418 |
| 19.4.2 本阶段页面概述..... | 374 | | |

| | | | |
|---|------------|-----------------------|------------|
| 20.3.1 本阶段迭代的目标 | 418 | 21.1.3 购书系统业务建模 | 468 |
| 20.3.2 本阶段页面概述 | 419 | 21.1.4 项目功能模块分割 | 468 |
| 20.3.3 用户以及用户空间的数据模型 | 424 | 21.2 迭代一用户登录, 书籍浏览 | 469 |
| 20.3.4 实现步骤 | 425 | 21.2.1 本阶段迭代目标 | 469 |
| 20.4 迭代三添加消息处理, 备忘录管理 | 441 | 21.2.2 本阶段页面概述 | 470 |
| 20.4.1 本阶段迭代的目标 | 441 | 21.2.3 本阶段数据模型 | 472 |
| 20.4.2 本阶段页面概述 | 442 | 21.2.4 实现步骤 | 474 |
| 20.4.3 消息和备忘录的数据模型 | 447 | 21.3 迭代二用户购书 | 485 |
| 20.4.4 实现步骤 | 447 | 21.3.1 本阶段迭代目标 | 485 |
| 20.5 迭代四添加管理员功能 | 457 | 21.3.2 本阶段页面概述 | 486 |
| 20.5.1 本阶段迭代的目标 | 457 | 21.3.3 实现步骤 | 487 |
| 20.5.2 本阶段页面概述 | 457 | 21.4 迭代三管理员登录, 用户信息查看 | 492 |
| 20.5.3 管理员的数据模型 | 459 | 21.4.1 本阶段迭代目标 | 492 |
| 20.5.4 实现步骤 | 459 | 21.4.2 本阶段页面概述 | 493 |
| 20.6 小结 | 466 | 21.4.3 本阶段数据模型 | 496 |
| | | 21.4.4 实现步骤 | 496 |
| 第 21 章 基于 Struts 2 和 Ajax 的网上购书系统 (Struts 2+Hibernate+Ajax+FreeMarker) | 467 | 21.5 迭代四管理员添加书籍以及添加用户 | 505 |
| 21.1 项目概述 | 467 | 21.5.1 本阶段迭代目标 | 505 |
| 21.1.1 购书系统概况 | 467 | 21.5.2 本阶段页面概述 | 505 |
| 21.1.2 购书系统需求分析 | 468 | 21.5.3 实现步骤 | 509 |
| | | 21.6 小结 | 513 |

第 1 篇 Struts 2 快速入门

第 1 章 Web 应用开发概览

第 2 章 第一个 Struts 2 应用

第 1 章

Web 应用开发概览

本章从 Web 开发的基础讲起，将 Web 开发的来龙去脉展现给各位读者。同时读者还将了解 Web 应用中采用 MVC 模式的根本原因以及 MVC 模式在 Web 开发框架中的发展现状，Web 应用开发的各种开源框架及其比较，Struts 2 高级特性介绍。几种流行的开源框架和 Struts 2 的比较，明确 Struts 2 的优缺点，使 Web 开发应用更具有针对性。

1.1 Web 应用与 Web 开发框架

1.1.1 Web 应用开发历程

Java 技术是目前流行的 Web 开发语言，特别是基于 J2EE 平台的 Java Web 开发模式已经被广泛使用，用户接触的很多动态网站都是采用了 Java 技术开发的。而 Java Web 开发技术的发展是和 Internet、浏览器的发展密不可分的。

Internet 是一个遵循一定协议自由发展的国际互联网，它利用覆盖全球的通信系统使各类计算机网络及个人计算机联通，从而实现智能化的信息交流和资源共享。

Internet 早已从最初学术科研网络变成了一个拥有众多商业用户、政府部门、机构团体和个人的综合的计算机信息网络。在发展规模上，目前 Internet 已经是世界上规模最大、发展最快的计算机互联网。TCP/IP 协议簇是目前 Internet 主流网络协议，它可提供任意互联的网络间的通信，几乎所用的网络操作系统都支持 TCP/IP 协议簇。在 TCP/IP 协议簇中，Web 服务以其使用的方便性占据了 TCP/IP 应用的绝大部分。Web 服务所采用的 HTTP 位于应用层，如同 TCP/IP 成为事实上的标准 Internet 网络协议一样，HTTP 所支持的 Web 应用成为 Internet 的最主要应用有一定的历史机遇和偶然性，而这种偶然性在很大程度上与浏览器的发展密不可分，浏览器的发展在一定程度上也决定了 Web 技术的发展。

提到 Web 浏览器，大多数人都会想到无处不在的微软公司的 Internet Explorer，简称 IE，直到最近像 Firefox、Safari 和 Opera 之类的浏览器推出，这种情况才稍有改观。1993 年，堪

萨斯大学开发人员编写了一个基于文本的浏览器，叫做 Lynx。1994 年，挪威奥斯陆的一个小组开发了 Opera，到 1996 年这个浏览器得到了广泛使用。1994 年 12 月，Netscape 发布了 Mozilla 的 1.0 版，第一个盈利性质的浏览器从此诞生。1996 年夏天，微软公司发布了 IE 3.0 版，几乎一夜之间人们纷纷奔向 IE。当时，因为 Netscape 的浏览器是要收费的，而微软公司则免费提供 IE。关于浏览器领域谁主沉浮，很多人担心 Microsoft 会像在桌面领域一样，在 Web 领域也一统天下。到 1999 年发布 IE 5.0 时，它已经成为使用最广泛的浏览器。2002 年，继承了 Netscape 的开源版本的 Firefox 浏览器异军突起，夺回了大量的市场份额。

Internet 提供了网络连接的基础架构，而浏览器则带给了 Internet 更为广泛的实际应用。下面让我们来深入了解一下影响浏览器发展的一个重要因素——Web 应用需求都经历了怎样的变化。回顾 Web 应用的发展历程及展望其光明前景，我们可以将其大致分为 3 个阶段：已经基本成为历史的 1.0 时代，正在日益发展的 2.0 时代和尚在萌芽中的 3.0 时代。

Web 1.0 时代，在计算机世界大行其道的是最先由 Borland 公司提出来的 C/S（客户机/服务器）结构模式。在这种模式下，服务器被启动，就随时等待响应客户程序发来的数据请求，这些请求通常只是一些数据库操作语句，当需要对数据库中的数据进行任何操作时，客户程序就自动地寻找服务器程序，并向它发出请求，服务器程序根据预定的规则作出应答，返回结果。

C/S 模式首次将程序和数据彻底分离，数据可以为不同程序所用，性能比文件服务器结构要强，维护起来比文件服务器结构容易，利用网络所带来的数据库的数据查询维护便利。然而，这种胖客户机/瘦服务器的模式使得资源没有合理使用，客户机程序配置复杂，开发难度大，难以使应用系统动态适应企业日益增长的业务需求，开发缺乏标准，系统的可用性和性能受到怀疑，同时也不便于异构系统的互连和扩充，难以保证应用系统的安全性。

C/S 模式的缺陷和不足很快就被微软公司注意到了，微软公司对 Borland 公司的 C/S 模式进行了小幅度的改动，然后提出了自己的 B/S 模式。这种 B/S 模式一般采用三层结构。

- 客户层 (browser)：用户接口和用户请求的发出地，典型应用是网络浏览器。
- 服务器层 (Web server)：典型应用是 Web 服务器和运行业务代码的应用程序服务器。
- 数据层 (data tier)：典型应用是关系型数据库和其他后端数据资源等。

B/S 模式极大地统一并简化了客户端，使用户的操作变得异常简单和一致。B/S 结构是对 C/S 结构的一种改进，用户工作界面是通过浏览器来实现的，浏览器根据获得的极少部分事务逻辑在前端 (Browser) 实现，主要事务逻辑在服务器端 (Server) 实现。这样就大大简化了客户端电脑载荷，减轻了系统维护与升级的成本和工作量，降低了用户的总体成本。然而，B/S 模式的三层结构只是简单把数据层分开，没有把程序代码按照逻辑分开，代码重用难度大，显示逻辑和业务功能逻辑没有分开，耦合性和移植性都比较差。

无论是 C/S 还是 B/S 结构都存在这样一些缺点：数据安全性低，对服务器要求过高，数据传输处理速率低等。例如，通过浏览器进行大量的数据输入或进行报表的应答，专用性打印输出都比较困难和不便，由此我们引出了一种分层的 Web 应用。它的体系结构如图 1-1 所示。

- 把显示逻辑从业务逻辑中分离出来，这就意味着业务代码是独立的，可以不关心怎样显示和在哪里显示。
- 业务逻辑层处于中间层，不需要关心由哪种类型的客户来显示数据，也可以与后端系统

保持相对独立性，有利于系统扩展。

- 多层结构具有更好的移植性，可以跨不同类型的平台工作，允许用户请求在多个服务器间进行负载均衡。

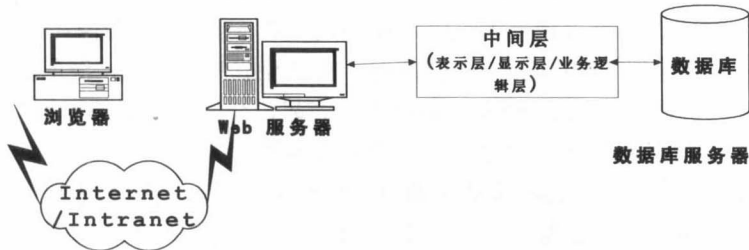


图 1-1 分层 Web 应用体系结构

分层的 Web 应用由于以上优势迅速成为 Web 应用的主体，其间动态网页由于交互性受到了广泛的关注并得到了长足的发展。目前，常用的 Web 动态网页开发技术有以下几种：CGI、ASP、PHP、JSP。这些动态网页的工作原理大致相同，简单归纳如下。

1. 当用户请求一个*.asp (*.jsp、*.php、*.cgi 等) 页面时，Web 服务器响应 HTTP 请求，调用 ASP (JSP、PHP、CGI 等) 引擎，解释 (编译) 被申请的文件。
2. 若脚本中含有访问数据库的请求，就通过 ODBC 或者 (JDBC) 与后台数据库相连，由数据库访问组件执行访问操作。
3. ASP 脚本在服务器端解释 (JSP 在服务器端是编译) 执行，根据访问数据库的结果集自动生成符合 HTML 语言的主页，去响应用户的请求。所有相关的发布工作由 WWW 服务器负责。

表 1-1 PHP、ASP、JSP 3 种动态 Web 页面技术比较

| Web 技术 | PHP | ASP | JSP |
|----------------|----------|------------------|---------------|
| 操作系统平台 | 所有操作系统 | Windows | 所有操作系统 |
| 开发语言 | PHP 脚本语言 | VBScript/JScript | Java |
| 使用组件 | 不支持 | COM/DCOM | JavaBean, EJB |
| 代码执行情况 | 解释执行 | 解释执行 | 一次编译 |
| 代码重用情况 | 不支持 | 支持 | 支持的很好 |
| 访问数据库组件 | 没有 | ODBC | JDBC |
| 支持的 Web Server | apache | IIS | 很多服务器 |
| 商家支持情况 | 没有 | 微软公司 | SUN |

除了这些纷繁的动态页面生成技术，丰富的 Web 页面表现技术也不断涌现。在过去的几年里经过长足发展的页面技术有 Applet、JavaScript 和 Ajax 等，正是这些技术使得 Web 应用更加动态，更加智能，并且更具表现力。

Web 1.0 是以信息的增值服务，围绕信息的搜集、获取、整理和发布而展开的。它的发展忽视了用户的能动性，用户只能浏览由网页编辑搜集整理的信息，而不能参与其中并加入讨论。网站对于用户来说形同生人。这样的一种模式不仅无法长时间地吸引用户，并且网站

形式及网站内容极易被其他网站和个人所复制，发展也受到人员和资金的限制。用户需要一种主观的投入，能够与网站展开互动，网站所需要做的事情就是发掘用户的需求并满足用户的需求，把复杂的技术开发留给自己，而把简单的使用体验交给用户，伴随着用户需求的改变，Web 2.0 就应运而生了。

一般来说，Web 2.0（也有人称之为互联网 2.0）是相对 Web 1.0 新的一类互联网应用的统称。Web 2.0 跟 Web 1.0 的最大区别在于没有了编辑，也就没有了信息控制，体现了一种“去中心化”的趋势。更注重用户的交互作用，用户既是网站内容的消费者，也是网站内容的制造者。Blogger Don 在他的“Web 2.0 概念诠释”一文中提到：Web 2.0 是以 Flickr、Craigslist、Linkedin、Tribes、Ryze、Friendster、43Things.com 等网站为代表，以 Blog、TAG、SNS、RSS、wiki 等社会软件的应用为核心，依据六度分隔、XML、Ajax 等新理论和技术实现的互联网新一代模式。

如果说 Web 1.0 时代的标准语言是 HTML，那么 Web 2.0 时代的标准语言就是 XML (eXtensible Markup Language, 可扩展标记语言)。这个“下一代网络应用的基石”自从它被提出来时就几乎得到了业界所有大公司的支持，丝毫不逊于当年 HTML 被提出来的热度。

1998 年 2 月，W3C (World Wide Web Consortium) 正式公布了 XML 的 1.0 版语法标准。XML 功能强大的主要原因在于 XML 是一种“元语言”(meta-language)。换言之，XML 是一种用来定义其他语言的语法系统。随着 XML 的发展，W3C 开发了一系列技术来规范和促进 XML 的发展。这一系列技术包括 DTD、XML Schema(XMLS)、RELAX NG DOM、SAX、Xpath XSL、XSLT、XSL-FO、CSS XLink、XPointer、Xquery 等。

如果说 Web 1.0 体现的网站和用户之间是一种陌生人的关系，Web 2.0 体现的网站和用户之间是主人和客人的关系的话，那么，到了 Web 3.0 时代，每一个用户都是 Web 的主人。

2005 年圣诞节，Bill.Gates 讲述了公司的互联网战略，主要围绕一个互联网新的概念模式展开，并给这种互联网模式一个新的名词 Web 3.0。从目前看来，未来的 Web 3.0 主要包含以下 3 个基本特征。

1. 网站内的信息可以直接和其他网站相关信息进行交互和转换，能通过第三方信息平台同时对多家网站的信息进行整合使用。
2. 用户在互联网上拥有自己的数据，并能不同网站上使用。
3. 完全基于 Web，用浏览器即可以实现复杂的系统程序才具有的功能。

1.1.2 为什么需要 Web 框架

Web 应用开发从 Internet 诞生以来就一直在发展着，从传统的 CGI 到更加高效的 Servlet，其间经历了无数技术上的变革。回顾 Web 的发展历史不难发现，实现高效的用户与服务器之间或者用户与用户之间的交互可以说就是 Web 应用的本质，这种交互需要技术上的不断进步来支持。

Web 层作为三层架构中的核心层，承担着 Web 应用的核心功能，在具备较大灵活性的同时，也不可避免地带来一些开发上的制约。由于技术发展的历史原因，虽然目前主流的 Web 应用开发工具 (PHP、JSP、ASP) 都提供了服务器端的页面展示、业务逻辑处理和数据访问等功能，但是这些功能并没有形成一个完整的整体，三层架构中 Web 层的混乱程度几乎到了难以忍受的地步，代码编写困难，调试更加困难。

让我们把注意力集中到 Web 层，看看 Web 层开发做了些什么工作。Web 层首先进行数据展示，从逻辑层获取需要展示的数据，以合理的方式在页面进行展示。接着接收用户从界面上输入的数据，比如接收界面上的按钮单击事件，驱动后台进行数据交换。然后调用逻辑层接口，根据收到用户的事件请求去调用相应的逻辑层接口来进行处理，组织成为逻辑层接口需要的数据封装形式。最后根据逻辑层的数据产生合适的页面展示给用户。

下面再来总结一下不使用任何开发框架进行 Web 层开发的大致步骤。

1. 编写 HTML 页面，将数据表现在页面上并指定具体表现形式，比如下拉列表或单选按钮等。进行界面表现形式的逻辑布局，比如数据表现的上下前后关系等。

2. 完成页面的表现形式后，需要进行一些功能性的开发。比如下拉列表中显示值的来源，开发人员需要编写代码来获取这些值，然后把这些值赋值到对应的表现形式中。

3. 开发功能性的事件响应。用户单击某个按钮或者触发了某个事件，服务器端获取到客户端提交的数据，然后调用相应的逻辑层接口来响应。

4. 编写逻辑层，实现相对应的业务逻辑功能。

5. 逻辑层执行完后，返回数据和信息到 Web 层，开发人员编写代码完成显示包含这些数据和信息的页面给用户。

6. 在整个交互的过程中，还必须考虑到诸多问题。比如数据访问权限、消息配置和国际化等。这些功能都起源于逻辑层，但是实际的控制 Web 层，都需要开发人员来控制。

7. 完成上面的开发步骤后，还需要把各个页面有机地组织起来，开发应用程序的整体应用导航框架，把各个功能页面与菜单结合起来，形成一个完整的应用。

在这里还省略了开发期间反复的调试过程。从以上开发过程中不难看出进行 Web 应用开发的复杂性，而这种复杂性也正是采用 Web 开发框架的根本原因。

本书主要讨论基于 Java 的 JSP/Servlet 的 Web 应用开发。下面让我们来看两个基于 JSP/Servlet 的 Web 应用示例程序，一方面回顾一下基于 JSP/Servlet 的核心概念，另一方面也感触一下采用 Web 开发框架的必要性。

Servlet & JSP 示例

helloworld.jsp:

```
<HTML>
  <HEAD>
    <TITLE>JSP 测试页面---HelloWorld!</TITLE>
  </HEAD>
  <BODY>
    <%
      <!-- 通过嵌入的 Java 代码打印一句 Html 语句 -->
      out.println("<h1>Hello World!<br>世界，你好! </h1>");
    %>
  </BODY>
</HTML>
```

Web.xml:

```
<Web-app xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
  http://java.sun.com/xml/ns/j2ee/Web-app 2.4.xsd"
  version="2.4">
  <description>
```