

Struts与 Hibernate 实用教程



—— 构建基于MVC模式的高效Java Web应用

邬继成 编著



随书附带光盘
包含本书重点范例
源代码和所用开源
软件的最新版本安装文件



电子工业出版社
Publishing House of Electronics Industry
<http://www.phei.com.cn>

Struts 与 Hibernate 实用教程

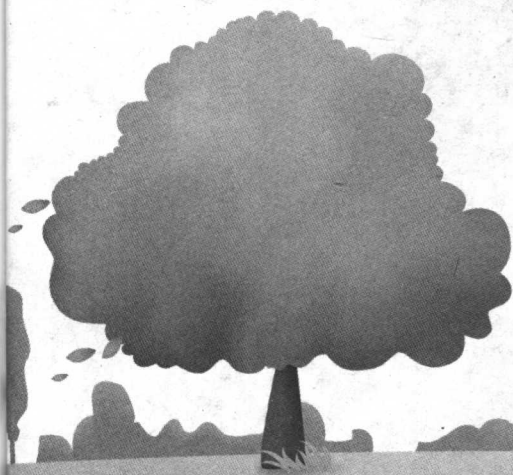
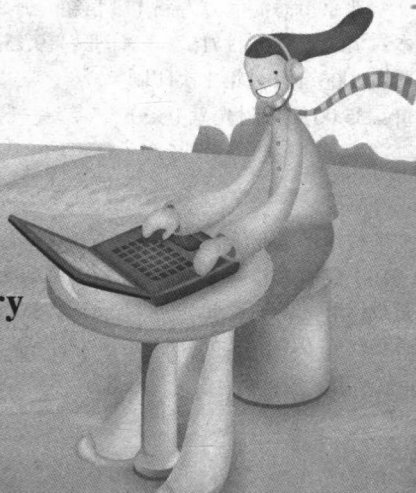
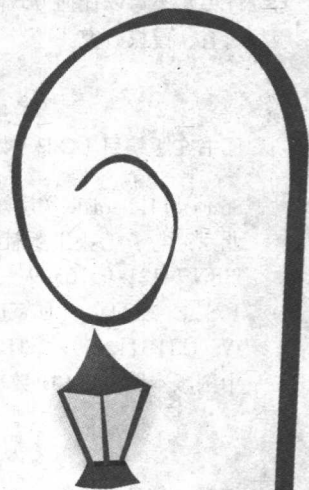
—— 构建基于 MVC 模式的高效 Java Web 应用

邬继成 编著

電子工業出版社

Publishing House of Electronics Industry

北京 · BEIJING



内 容 简 介

Struts 和 Hibernate 是当前非常流行的 Java Web 应用框架, 由于它们很好地实现了 MVC 设计模式以及具有使用简便、开源免费的特点, 在国内外获得越来越广泛的应用。本书将结合实际例子由浅入深地介绍 Struts 和 Hibernate 的基本原理和应用方法, 内容包括 Java Web 应用基础, Struts 和 Hibernate 的基本原理, 实例讲解 Struts 和 Hibernate 应用程序的开发方法, 一些相对高级但实用的 Struts 和 Hibernate 技术, 以及 Struts 和 Hibernate 结合在一起使用来构建一个完整的基于 MVC 模式的 Java Web 应用程序。

本书内容安排采用实用人才培养的思路, 由浅入深, 实用为主, 既可以供广大工程技术人员参考, 也是各类院校学生学习 Java Web 开发的绝佳入门读本。

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

Struts 与 Hibernate 实用教程: 构建基于 MVC 模式的高效 Java Web 应用 / 邹继成编著.

北京: 电子工业出版社, 2006.9

ISBN 7-121-03163-9

I. S... II. 邹... III. ①软件工具 - 程序设计 - 教材 ②Java 语言 - 程序设计 - 教材

IV. ①TP311.56 ②TP312

中国版本图书馆 CIP 数据核字 (2006) 第 106778 号

责任编辑: 贺瑞君

印 刷: 北京市天竺颖华印刷厂

装 订: 三河市金马印装有限公司

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 × 980 1/16 印张: 19.75 字数: 506 千字

印 次: 2006 年 9 月第 1 次印刷

定 价: 35.00 元 (附光盘 1 张)

凡所购买电子工业出版社的图书有缺损问题, 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, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线: (010) 88258888。

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

随着互联网和浏览器技术的普及，Web 编程已变得越来越重要，各种 Web 编程技术和框架也随之涌现出来。在 Web 编程中，MVC (Model-View-Controller，即模型、视图和控制器) 设计模式已经成为一种最主要的设计模式。Struts 作为第一个基于 MVC 设计模式的开源 Java Web 编程框架，在业界受到了广泛的尊崇，得到了普遍应用，已经成为一个主流的 Web 编程框架。

任何应用软件都要与数据库连接，这就存在一个持久性问题，即数据在数据库中的存取问题。为解决这个问题，J2EE 推出了实体 Bean (Entity Bean)。但在后来的应用中，人们发现实体 Bean 存在许多问题，使用很不方便，不容易掌握，而且一旦程序出现 Bug，很难找到原因，不容易调试。而 Hibernate 成功地解决了持久性问题。作为一个开源软件，它非常容易使用，可以根据用户的需要进行修改。所以它一经推出，立刻受到了业界空前的欢迎。

目前市面上已有一些关于 Struts 和 Hibernate 的图书，但它们许多都是大部头专著，过于理论化和系统化，其中很多内容应用很少。初学者要读完这样的书，需要耗费很长的时间和很多的精力，而且读完后往往还不知道如何着手编写 Struts 和 Hibernate 程序。因此，许多这样的大部头并不适合初学者。

本书内容来自作者多年的教学和软件开发经验，主要面对初学者，特别是针对面对就业压力、需要实践技能的大学生和年轻的程序开发爱好者。本书也可用做大专院校和计算机培训学校的指导图书。

本书的内容可分为三部分：第一部分讲解 Struts 的原理及其应用；第二部分讲解 Hibernate 的原理和应用；第三部分是提高篇，介绍如何用 Struts 和 Hibernate 一起构建 Java Web 应用，包括在 Eclipse 中开发 Struts 和 Hibernate 应用的方法，同时涉及一些常用开源软件工具的使用法。

本书将结合应用例子由浅入深地介绍 Struts 和 Hibernate 的基本原理，着重于它们的实际应用。掌握本书内容后，读者将能够自己编写 Struts 和 Hibernate 的应用程序。因为把 Struts 和 Hibernate 结合在一起使用，可以构建出高效、完整的 Java Web 应用，所以作者把它们写入一书中，这样可以突出它们之间的密切联系，有助于培养读者把这两种工具结合使用的良好习惯。如果将它们分别写成专著，则容易割裂它们之间的关系。一句话，该书是一本强调 Struts 和 Hibernate 开发技能的实际指导图书。我深信，该书会对将要走上工作岗位的广大学子和程序开发爱好者有实质性的帮助。如对本书有任何建议和批评，欢迎与我联系。我的电子邮箱是 jcwul67@126.com。

邬继成

2006 年 9 月于北京

目 录

第 1 章	Java Web 编程基础	1
1.1	Java Web 编程的主要组件技术	1
1.1.1	Servlet	1
1.1.2	JSP	5
1.1.3	JavaBean	9
1.1.4	JDBC	10
1.1.5	XML	14
1.1.6	Tomcat	14
1.2	MVC 设计模式	17
1.2.1	JSP Model 1 和 Model 2 架构	17
1.2.2	MVC 设计模式	17
1.2.3	MVC 实现框架	18
1.3	构建一个简单的基于 MVC 模式的 Java Web 应用程序	18
1.3.1	数据表设计	19
1.3.2	构建视图组件	19
1.3.3	构建控制组件	21
1.3.4	构建模型组件	23
1.3.5	构建数据访问组件	24
1.3.6	编译、打包、部署和运行程序	26
1.3.7	讨论	28
1.4	小结	28
第 2 章	Struts 入门	29
2.1	Struts 简介	29
2.1.1	Struts 软件包的下载和安装	29
2.1.2	Struts 软件包的组成	29
2.2	Struts 的基本原理	30
2.3	Struts 的核心组件	31
2.3.1	Struts 的控制器组件	31
2.3.2	Struts 的视图组件	33
2.3.3	Struts 的模型组件	36

2.4	用 Struts 构建一个简单的登录系统	37
2.4.1	用 JBuilder 建立一个项目工程	37
2.4.2	构建 JSP 页面	37
2.4.3	构建 ActionForm	41
2.4.4	构建 Action	45
2.4.5	构建模型组件和数据访问组件	48
2.4.6	构建 Struts 的配置文件	48
2.4.7	编译、打包、部署和运行程序	50
2.5	小结	51
第3章	Struts 详解	52
3.1	Struts 配置文件 struts-config.xml	52
3.1.1	<form-beans>元素	53
3.1.2	<action-mappings>元素	54
3.1.3	<global-forwards>元素	55
3.1.4	<message-resources>元素	55
3.2	Struts 的中央控制器	56
3.2.1	Struts 应用程序处理用户请求的一般过程	56
3.2.2	ActionServlet 类	57
3.2.3	RequestProcessor 类	58
3.2.4	ActionServlet 在 web.xml 中的配置	60
3.3	Action 类及其相关类	60
3.3.1	Action 类	60
3.3.2	ActionMapping 类	62
3.3.3	ActionForward 类	63
3.4	ActionForm 类及表单数据验证	64
3.4.1	Struts 的视图组件概述	64
3.4.2	ActionForm 的作用机理	64
3.4.3	ActionForm 的使用方法	66
3.4.4	表单数据验证	68
3.5	Struts 常用标记库	71
3.5.1	HTML 标记库	71
3.5.2	Bean 标记库	77
3.5.3	Logic 标记库	80
3.6	Struts 应用示例	84
3.7	小结	94

第 4 章 Struts 提高	95
4.1 Struts 对国际化的支持	95
4.1.1 资源文件和资源包	95
4.1.2 资源文件的编码转化	97
4.1.3 资源包的访问	98
4.1.4 国际化的应用举例	99
4.2 DispatchAction 类	101
4.3 动态 ActionForm	103
4.3.1 配置动态 ActionForm	103
4.3.2 在 Action 中访问动态 ActionForm	104
4.3.3 动态 ActionForm 的表单验证	105
4.3.4 动态 ActionForm 应用实例	105
4.4 Validator 验证框架	106
4.4.1 Validator 的安装	107
4.4.2 在 struts-config.xml 中配置 Validator	107
4.4.3 validator-rules.xml 的配置	107
4.4.4 validation.xml 的配置	109
4.4.5 DynaValidatorForm 类及其子类	113
4.4.6 Validator 的应用示例	114
4.5 小结	118
第 5 章 Hibernate 入门	119
5.1 Hibernate 简介	119
5.2 Hibernate 软件包简介	120
5.3 Hibernate 框架简介	121
5.3.1 Hibernate 的结构体系	121
5.3.2 Hibernate 的核心组件	122
5.3.3 Hibernate 的运行过程	123
5.4 Hibernate 入门示例	123
5.5 小结	132
第 6 章 Hibernate 详解	133
6.1 Hibernate 配置文件	133
6.1.1 hibernate.properties	133
6.1.2 hibernate.cfg.xml	136
6.2 持久化类 / 对象	137
6.3 映射文件 xxx.hbm.xml	139

6.4	Configuration 类	142
6.5	SessionFactory 接口	143
6.6	Session 接口	144
6.6.1	概述	144
6.6.2	取得持久化对象的方法	145
6.6.3	持久化对象的保存、更新和删除方法	146
6.7	Query 接口	150
6.7.1	概述	150
6.7.2	setXXX()方法	150
6.7.3	list()方法	151
6.7.4	excuteUpdate()方法	151
6.7.5	使用命名查询 (namedQuery)	152
6.8	Transaction 接口	153
6.9	HibernateUtil 类	154
6.10	Hibernate 应用示例	155
6.11	小结	160
第 7 章	Hibernate 提高	161
7.1	利用关联关系操纵对象	161
7.1.1	一对一关联关系的使用	161
7.1.2	一对多关联关系的使用	166
7.1.3	多对多关联关系的使用	168
7.2	Hibernate 数据查询	170
7.2.1	Hibernate Query Language	170
7.2.2	Criteria Query 方式	174
7.2.3	Native SQL 查询	176
7.3	Hibernate 的事务管理	179
7.3.1	事务的特性	179
7.3.2	事务隔离	180
7.3.3	在 Hibernate 配置文件中设置隔离级别	181
7.3.4	在 Hibernate 中使用 JDBC 事务	182
7.3.5	在 Hibernate 中使用 JTA 事务	183
7.4	Hibernate 的 Cache 管理	184
7.4.1	一级 Cache	184
7.4.2	二级 Cache	185
7.4.3	在 Hibernate 中使用 EhCache	187
7.5	小结	188

第 8 章	用 Struts 和 Hibernate 一起构建 Java Web 应用	189
8.1	用 Struts 和 Hibernate 构建一个列车车次查询系统	189
8.1.1	设计数据表	189
8.1.2	构建视图组件	190
8.1.3	构建 Struts 组件	192
8.1.4	构建模型层组件	194
8.1.5	构建 Hibernate 组件	196
8.1.6	编译、打包与运行	199
8.2	用 Struts 和 Hibernate 构建一个在线招聘系统	202
8.2.1	需求说明和分析	202
8.2.2	招聘岗位管理模块的编程	205
8.3	自己动手	223
第 9 章	在 Eclipse 中开发 Struts 和 Hibernate 应用	224
9.1	Eclipse 概述	224
9.1.1	Eclipse 的由来和发展	224
9.1.2	Eclipse 的结构和平台内核	224
9.2	Eclipse 开发环境的建立	225
9.2.1	Eclipse 安装	225
9.2.2	Eclipse 多国语言包的安装	227
9.2.3	Eclipse 中文文本编辑器的设置	228
9.3	在 Eclipse 中进行 Java Web 应用开发	229
9.3.1	Eclipse 中 Java Web 应用开发环境的建立	229
9.3.2	在 Eclipse 中开发 Java Web 应用	232
9.4	在 Eclipse 中进行 Struts 应用开发	245
9.4.1	Easy Struts 插件简介	245
9.4.2	Easy Struts 插件的安装与配置	246
9.4.3	开发 Struts 应用程序	247
9.5	在 Eclipse 中进行 Hibernate 应用开发	267
9.5.1	Hibernate Synchronizer 简介	267
9.5.2	Hibernate Synchronizer 的下载和安装	268
9.5.3	Hibernate 应用程序的开发	268
9.6	小结	280
第 10 章	Java Web 应用开发中常用的开源软件工具	281
10.1	开源软件概述	281
10.1.1	开源软件的历史与现状	281

10.1.2	主要开源软件项目介绍	282
10.1.3	基于开源软件的 Java Web 应用开发的技术方案	284
10.2	构建工具 Ant 的用法	284
10.2.1	Ant 简介	284
10.2.2	Ant 的安装与配置	285
10.2.3	Ant 的构建文件 build.xml	286
10.2.4	编译源代码	286
10.2.5	文件系统操作	288
10.2.6	应用举例	289
10.3	Log4j 的用法	291
10.3.1	Log4j 简介	291
10.3.2	Log4j 的组成	291
10.3.3	Log4j 的配置	296
10.3.4	Log4j 的应用实例	297
10.4	MySQL 的用法	300
10.4.1	MySQL 数据库简介	300
10.4.2	MySQL 的下载与安装	301
10.4.3	MySQL 的常用命令	301
10.5	小结	305

第1章 Java Web 编程基础

随着互联网和浏览器技术的日益普及，Web 应用编程的重要性逐渐凸现出来，各种 Web 编程技术和框架也随之涌现出来。本章将简单介绍一些主流的 Java Web 编程技术、设计模式和框架。

本章的主要内容如下：

- 简单介绍 Java Web 编程的主要组件技术。
- 介绍 MVC 设计模式。
- 结合实例介绍如何构建一个基于 MVC 模式的 Java Web 应用程序。

通过本章的学习，大家应该对 Java Web 应用开发有一个比较全面的了解，并掌握如何开发一个简单的基于 MVC 设计模式的 Java Web 应用。

1.1 Java Web 编程的主要组件技术

一个完整的 Java Web 应用软件通常是由多种组件构成的，一般由表示层组件、控制层组件、业务逻辑层组件及数据访问层组件组成。表示层组件通常由 HTML 和 JSP 页面构建，控制层组件一般是 Servlet，业务逻辑层组件是 JavaBeans 或 EJB，数据访问层组件是 JDBC，Hibernate 或 CMP。此外，Java Web 应用的各个组件需要在 XML 格式的配置文件中进行声明，然后打包，部署到 Java Web 服务器（例如 Tomcat）中运行。下面我们将分别介绍 Servlet，JSP，JavaBeans，JDBC，XML 和 Tomcat 等主要的 Java Web 组件技术。

1.1.1 Servlet

Servlet（Java 服务器小程序）是用 Java 编写的服务器端程序，是由服务器端调用和执行的、按照 Servlet 自身规范编写的 Java 类。

Servlet 可以处理客户端传来的 HTTP 请求，并返回一个响应。它是一个 Java 类，具有可移植、功能强大、安全、集成、模块化和可扩展性好等特点。代码清单 1.1 所示为一个简单的小服务器程序。

代码清单 1.1 一个简单的 Servlet 例子

```
-----  
import javax.servlet.*;  
import java.io.* ;  
  
public class HelloWorld extends GenericServlet
```

```
{
    public void service(ServletRequest request, ServletResponse response)
        throws IOException
    {
        // 设定 Web 服务器的响应方式是 HTML
        response.setContentType("text/html");

        // 获得 PrintWriter 对象, 以打印输出响应
        PrintWriter out = response.getWriter();

        // 把 HTML 形式的响应显示在浏览器上
        out.println("<HTML>");
        out.println("<BODY>");
        out.println("<H1>Hello World!<H1>");
        out.println("</BODY>");
        out.println("</HTML>");
    }
}
```

----- End -----

Servlet 需要部署在 Web 容器中, 它的生命周期由 Web 容器管理, 可以分为以下几个阶段:

- 装载 Servlet: 这项操作一般是动态执行的。有些服务器提供了相应的管理功能, 可以在启动的时候就装载 Servlet, 并能够初始化特定的 Servlet。
- 创建一个 Servlet 实例。
- 调用 Servlet 的 `init()` 方法。
- 服务: 如果 Web 容器接收到对此 Servlet 的请求, 那么它调用 Servlet 的 `service()` 方法。
- 销毁: 销毁实例, 通过调用 Servlet 的 `destroy()` 方法来销毁 Servlet。

Java 对 Servlet 的支持是通过 `javax.servlet` 和 `javax.servlet.http` 两个软件包实现的, 其中 `javax.servlet.http` 是 `javax.servlet` 的一个子软件包。

`javax.servlet` 软件包包含 Servlet 常用的接口和类, 如 `GenericServlet`, `ServletConfig`, `ServletRequest`, `ServletResponse` 和 `RequestDispatcher` 等。

`javax.servlet.Servlet` 接口提供了创建 Servlet 的一般框架。Servlet 可直接实现此接口, 或者间接地通过扩展 `javax.servlet.GenericServlet` 类或 `javax.servlet.http.HttpServlet` 类来实现此接口。`javax.servlet.GenericServlet` 类可用于创建与任何协议一起使用的 Servlet。

而 `javax.servlet.http` 软件包除了继承以上接口和类外, 还包含一些针对 HTTP 协议的方法、特殊请求和响应对象, 如 `HttpServlet`, `HttpServletRequest`, `HttpServletResponse`, `HttpSession` 和 `HttpDispatcher` 等。下面我们对它们进行简单介绍。

- **HttpServlet**类提供了处理不同HTTP请求类型（有GET、POST和PUT等）的方法，其中两种最常见的HTTP请求类型是GET和POST，它们分别是由doGet()和doPost()方法进行处理：

```
protected void doGet(HttpServletRequest request,
                    HttpServletResponse response)
    throws ServletException, java.io.IOException;
```

```
protected void doPost(HttpServletRequest request,
                    HttpServletResponse response)
    throws ServletException, java.io.IOException;
```

doGet()和doPost()方法与代码清单1.1中的service()方法具有相同的形式。**HttpServlet**类的doGet()与doPost()方法检查HTTP请求，然后调用相应的处理方法处理此请求。

- **HttpServletRequest**接口提供了处理客户请求的方法，如String getParameter (String name), String[] getParameterValues (String name)等方法。使用这些方法，Servlet可从客户页面窗体中取得数据。例如，从客户页面窗体取得用户名（username）的方法如下：

```
HttpServletRequest request;
String name = request.getParameter("username");
```

- **HttpServletResponse**接口提供了以HTML页面形式把请求发送给客户的方法，主要方法有setContentType(), sendRedirect()等。例如，把请求转发给下一个页面next.jsp的实现方法如下：

```
HttpServletResponse response;
response.sendRedirect("next.jsp");
```

- **HttpSession**接口用来记录当前Servlet中的用户会话。例如，在Web站点上注册的每个用户自动与HttpSession对象关联。Servlet可用此对象来存储关于用户会话的信息。可使用HttpSession接口的putValue()和getValue()等方法把数据存入HttpSession对象中，或从该对象中取出数据。通过HttpServletRequest对象的getSession()方法可生成一个HttpSession对象，如下面的代码所示：

```
HttpServletRequest request;
HttpSession session = request.getSession(true);
```

- **RequestDispatcher**接口可以把Servlet的请求提交或委派给另一个资源，如Servlet，HTML文件或JSP页面。在这种情况下，源Servlet进行同样处理并把请求委派给另一个Servlet。**RequestDispatcher**接口封装指向特定资源（Servlet，HTML文件或JSP页面）的URL。用此接口的forward (HttpServletRequest request, HttpServletResponse response)方法委派任务到

特定资源。RequestDispatcher 对象可通过 HttpServletRequest 对象的 getRequestDispatcher() 方法生成。例如：

```
RequestDispatcher rd = request.getRequestDispatcher("reply.jsp");  
rd.forward(request, response);
```

代码清单 1.2 是一个典型的 HttpServlet 应用例子。

代码清单 1.2 HttpServlet 的应用

```
-----  
import javax.servlet.http.*;  
import javax.servlet.*;  
import java.io.*;  
import java.util.*;  
  
public class MyFirstServlet extends HttpServlet{  
  
    public void doPost(HttpServletRequest request, HttpServletResponse response){  
  
        String name = request.getParameter("username");  
        RequestDispatcher rd;  
  
        try {  
            try {  
                if(name.equals("Tom")) {  
                    HttpSession session = request.getSession(true);  
                    session.setAttribute("nm", name);  
                    response.sendRedirect("reply.jsp");  
                }  
  
                else {  
rd = request.getRequestDispatcher("error.jsp");  
                rd.forward(request, response);  
                } } catch(ServletException e) { System.out.println(e);}  
            } catch(IOException e) { System.out.println(e);}  
        }  
  
        public void doGet(HttpServletRequest request, HttpServletResponse response){  
            doPost(request, response);  
        }  
    }  
}-----  
-----End -----
```

在 Java Web 应用程序中, Servlet 常用做控制器组件。当 Servlet 部署到 Web 服务器中时, 需要在 Web 配置文件 web.xml 中进行声明, 如代码清单 1.3 所示。

代码清单 1.3 web.xml

```
-----
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application
2.3//EN" "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>

    <servlet>
        <servlet-name>myFirstServlet</servlet-name>
        <servlet-class>MyFirstServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>myFirstServlet</servlet-name>
        <url-pattern>/myFirstServlet</url-pattern>
    </servlet-mapping>

</web-app>
-----
```

End

1.1.2 JSP

JSP (Java Server Page) 页面由 HTML 代码和嵌入其中的 Java 代码组成。在页面被客户端请求后, 服务器对这些 Java 代码进行处理, 然后将生成的 HTML 页面返回客户端的浏览器。Java 技术简单易用、完全面向对象、平台无关、安全可靠和面向 Internet 等特点, 都在 JSP 中得到了体现。一个简单的 JSP 页面如代码清单 1.4 所示。

代码清单 1.4 一个简单的 JSP 页面

```
-----
<html>
<body>
<%
    out.println("<H1>Hello World!</H1>");
%>
</body>
</html>
-----
```

End

JSP 页面一般包含 JSP 指令、JSP 脚本元素、JSP 标准动作及 JSP 隐式对象。

1.1.2.1 JSP 指令

JSP 页面中的指令元素提供了特定 JSP 页面的全局信息，可分为三种类型：page 指令，include 指令及 taglib（标记库）指令。把指令加入 JSP 页面的语法如下：

```
<%@ directive (attribute="attribute value") %>
```

1) page 指令

page 指令定义了一些属性，它们用来通知 Servlet 引擎有关 JSP 页面的一般设置。这些属性包括 language，contentType，import，session，autoFlush，errorPage 及 isErrorPage 等。它们的用法如表 1.1 所示。

表 1.1 page 指令

属性	描述和用法
language	在编写 JSP 页面时要用的脚本语言，例如 Java 语言。 用法：<%@ page language="java" %>
contentType	应答 MIME 类型（多用途 Internet 邮件扩充类型）。属性的默认值为 text/html。例如： <%@ page contentType="text/html; charset=GB2312" %>
import	可用来在 JSP 页面中引入软件包或类，例如： <%@ page import="java.io.*; java.util.*" %>
session	用来指出关于特定 JSP 页面的会话数据的可用性。此属性的默认值为 true。例如： <%@ page session="false" %>
autoFlush	自动刷新选项，如果是 true 则页面自动刷新，否则页面不自动刷新
errorPage	错误页面的 URL，用来显示错误。例如： <%@ page errorPage="/error.jsp" %>
isErrorPage	指定是否将当前页面用做 JSP 错误页面。如果是 true，则将当前页面用做 JSP 错误页面；如果是 false，则否。例如： <%@ page isErrorPage="true" %>

2) include 指令

include 指令用来指出编译 JSP 页面时要插入的文件名（以相对 URL 形式），所以被包括的文件内容成为 JSP 页面的一部分。include 指令也可以用来插入为多页面所公用的部分代码。例如，在一个 JSP 页面中包含头文件（header.html）的方法如下：

```
<%@ include file ="header.html" %>
```

3) taglib（标记库）指令

taglib 指令，或称为标记库指令，用来在 JSP 页面中引入该页面所要用的标记库资源。例如，在 Struts 中页面一般要用到标记库 struts-html.tld 中的标记，因此要在这些页面中引入该标记库，方法如下：

```
<%@ taglib prefix="html" uri=" /WEB-INF/struts-html.tld" %>
```