HZ BOOKS
华章教育

高等院校计算机教材系列

JSP 2.0大学教程

覃华 韦兆文 陈琴 编著

为教师提供电子教案等资源

机械工业出版社
China Machine Press

高等院校计算机教材系列

JSP 2.0大学教程

覃华　韦兆文　陈琴　编著

机械工业出版社
China Machine Press

本书通过案例全面讲解了JSP 2.0的基础知识，内容主要包括：JSP元素、JSP隐含对象与作用范围变量、Servlet程序设计、过滤器与侦听器、JavaBean组件技术、JDBC数据库访问技术、EL与JSTL标记库、Struts 1.3框架等，其中融入了VO、DAO、MVC等设计模式概念，整合了DreamWeaver 8 + JDK 1.5 + Tomcat 5.5 + JCreator 4开发工具。全书最后通过一个基于Struts 1.3的网上书店项目分析，讲解JSP应用系统的基本开发过程。此外，本书为教学配备了电子资源包，包括PPT电子教案、全部样例的源代码和部分典型例题的操作视频等。

本书内容全面，通俗易懂，案例可操作性强，既注重基础理论的讲解，又强调实践技能的培养。本书可作为高等院校相关专业的JSP课程教材，也可供JSP初学者自学或JSP技术培训用。

图书在版编目(CIP)数据

JSP 2.0大学教程/覃华等编著. —北京：机械工业出版社，2008.1
(高等院校计算机教材系列)

ISBN 978-7-111-22887-5

Ⅰ. J… Ⅱ. 覃… Ⅲ. JAVA语言－主页制作－程序设计－高等学校－教材 Ⅳ. TP393.092

中国版本图书馆CIP数据核字(2007)第185353号

机械工业出版社(北京市西城区百万庄大街22号　邮政编码　100037)
责任编辑：迟振春
北京瑞德印刷有限公司印刷 · 新华书店北京发行所发行
2008年1月第1版第1次印刷
184mm×260mm · 20印张
标准书号：ISBN 978-7-111-22887-5
定价：32.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换
本社购书热线：(010)68326294

前　言

随着 Internet 技术的普及和发展，越来越多的电子商务和电子政务系统采用 Web 架构实现。JSP 是由 Sun 公司倡导、许多大公司参与制定的一种动态网页技术标准，属于 J2EE 技术规范的组成部分之一。JSP 以 Java 语言为底层支持，可扩展为企业级应用，并且技术标准开放，这使得它逐渐成为 Web 系统开发的首选。

目前，越来越多的高等院校将 JSP 技术列为专业必修内容。本书是按照本科教学要求编写的，也适合高职高专学校使用。本书是从编者的讲义演变而来的，该讲义从 2002 年开始用于本科教学，在此后的多年教学过程中，我们根据教学积累、技术发展和项目经验，对讲义进行了多次修订，最终形成本教材。本书具有以下特色：

- 面向实践技能培养，采用案例教学法组织内容。对于关键的知识模块，提供一个典型的、可操作的实验案例，通过实例来增强读者对知识的理解，并有助于培养读者的学习兴趣和实践技能。
- 实用性和综合性强。在选取内容时，把实践项目中常用到的基础技术融入教材中，并将常用的 DAO、VO、MVC 等设计模式概念引入书中，最后综合全书技术，讲解了一个基于 Struts 1.3 的网上书店项目的实施过程。

本书共分 9 章。第 1 章介绍 JSP 2.0 和 Servlet 2.4 的概貌，以及如何建立上机实验环境。第 2 章讲解常用的 JSP 元素。第 3 章讲解 JSP 隐含对象的基本用法，以及 JSP 作用范围变量的含义和使用方法。第 4 章讲解 Servlet 程序的编程过程，并介绍过滤器和侦听器的基本用法。第 5 章讲解 JavaBean组件技术，介绍用 JavaBean 来封装业务逻辑和数据，并讲解了一些典型组件的使用方法。第 6 章讲解用 JDBC 对数据库进行增、删、改、查的基本方法，并介绍了连接池、预编译、JDBC 事务处理和数据库存储过程调用技术。第 7 章讲解 EL 表达式和 JSTL 1.1 标记库的用法，并讲解如何用 Tag File 制作自定义标记。第 8 章讲解 MVC 设计模式的思想和 Struts 1.3 的核心技术，并简要地介绍 Struts 1.3 的表单标记和表单校验技术。第 9 章综合全书技术，讲解一个基于 Struts 1.3 的网上书店系统的开发过程，重点介绍 Struts 中数据库查询与分页、购物车模块的设计思想。全书建议分 60 个多媒体课时讲授。

为了方便教学，本书配备了电子资源包，包括 PPT 电子教案、全部样例的源代码和部分典型例题的操作视频。电子资源包可从华章网站（http://www.hzbook.com）上下载。在学习本书的样例时，建议结合电子资源包中的样例源代码进行。

全书初稿由覃华老师编写，韦兆文、陈琴老师负责书稿的修订和审校，陈琴、曹波、闭剑婷、徐燕子、农佳捷、潘春孟、包远富、陈海军等参与了本书电子资源包的制作和后期审校。最后，全书由覃华老师统稿。本书是苏一丹教授主持的“十一五”教改项目内容之一，感谢苏老师对本书的支持和帮助。

由于编者水平有限，加上审校时间仓促，书中难免有错漏之处，敬请广大读者批评指正，我们会根据读者的意见适时修订教材。我们的联系邮箱是 qhgxu@126.com。

编　者

2007 年 10 月

目　录

前言

第 1 章　JSP 概述 …… 1
1.1　静态网页和动态网页 …… 1
1.1.1　静态网页 …… 1
1.1.2　动态网页 …… 2
1.2　Servlet 与 JSP …… 3
1.2.1　Servlet 技术 …… 3
1.2.2　JSP 概述 …… 4
1.3　上机实验环境的搭建 …… 4
1.3.1　安装 J2SE 1.5 …… 4
1.3.2　安装 Tomcat 5.5 …… 5
1.3.3　安装 DreamWeaver 8 简体中文版 … 10
1.3.4　安装 JCreator 4 …… 11
1.4　第一个 JSP 网页 …… 11
1.5　小结 …… 13
1.6　习题 …… 13
第 2 章　JSP 元素 …… 16
2.1　JSP 页面的组成元素与常用的 HTML 标记 …… 16
2.1.1　JSP 页面的组成元素 …… 16
2.1.2　常用的 HTML 标记 …… 17
2.2　JSP 注释元素 …… 21
2.2.1　JSP 注释 …… 21
2.2.2　HTML 注释 …… 21
2.3　JSP 指令元素 …… 21
2.3.1　include 指令 …… 22
2.3.2　page 指令 …… 22
2.3.3　taglib 指令 …… 26
2.4　JSP 脚本元素 …… 27
2.4.1　声明元素 …… 27
2.4.2　脚本小程序 …… 28
2.4.3　表达式元素 …… 29
2.5　JSP 标准动作 …… 30
2.5.1　JSP 2.0 的 20 种标准动作 …… 30
2.5.2　<jsp:include> 动作 …… 31
2.5.3　<jsp:param> 动作 …… 31
2.5.4　<jsp:forward> 动作 …… 32
2.5.5　<jsp:plugin> 动作 …… 32
2.5.6　<jsp:params> 和 <jsp:fallback> 动作 …… 33
2.5.7　XML 与 XML 文档 …… 34
2.6　小结 …… 35
2.7　习题 …… 36
第 3 章　JSP 隐含对象 …… 38
3.1　JSP 的隐含对象 …… 38
3.2　out 隐含对象 …… 39
3.2.1　输出信息的方法 …… 39
3.2.2　缓冲区相关的方法 …… 40
3.3　request 隐含对象 …… 41
3.3.1　用 request 读取客户端传递来的参数 …… 41
3.3.2　request 作用范围变量 …… 45
3.3.3　用 request 读取系统信息 …… 49
3.3.4　用 request 读取 HTTP 请求报头信息 …… 50
3.3.5　用 request 读取 cookie …… 51
3.3.6　用 request 选择国际化信息 …… 55
3.4　response 隐含对象 …… 55
3.4.1　输出缓冲区与响应提交 …… 55
3.4.2　HTTP 响应报头设置 …… 56
3.4.3　用 response 实现 JSP 页面重定向 … 59
3.4.4　用 response 实现文件下载 …… 60
3.5　application 隐含对象 …… 64
3.5.1　用 application 访问 Web 应用的初始参数 …… 64
3.5.2　application 作用范围变量 …… 65
3.5.3　用 application 对象读取 Servlet 容器信息 …… 67
3.5.4　用 application 记录操作日志 …… 67
3.5.5　application 的其他应用 …… 68
3.6　session 隐含对象 …… 69
3.6.1　用 URL 重写实现 session 跟踪 …… 69
3.6.2　用 cookie 实现 session 跟踪 …… 72

3.6.3 用隐藏表单域实现 session 跟踪…… 73
3.6.4 session 作用范围变量与 session 跟踪 …… 75
3.6.5 动态生成验证码 …… 79
3.7 其他 JSP 隐含对象 …… 81
3.7.1 config 隐含对象 …… 81
3.7.2 exception 隐含对象 …… 82
3.7.3 page 隐含对象 …… 82
3.7.4 pageContext 隐含对象 …… 83
3.8 小结 …… 84
3.9 习题 …… 84
第 4 章 Servlet、过滤器与侦听器 …… 87
4.1 Servlet 包的构成与 Servlet 生命周期 …… 87
4.1.1 GenericServlet 抽象类 …… 88
4.1.2 HttpServlet 抽象类 …… 88
4.1.3 Servlet 程序的生命周期 …… 89
4.2 Servlet 编程 …… 92
4.2.1 Servlet 程序的编写过程 …… 92
4.2.2 第一个 Servlet 程序 …… 96
4.2.3 Servlet 与 HTML 表单 …… 97
4.2.4 Servlet 通信 …… 99
4.2.5 Servlet 的 session 跟踪 …… 101
4.2.6 用 JSP 页面作 Servlet 程序 …… 104
4.3 过滤器 …… 106
4.3.1 过滤器的基本工作原理 …… 106
4.3.2 过滤器的 API 接口及部署信息 … 108
4.3.3 第一个过滤器程序 …… 110
4.3.4 用过滤器解决 request 中文乱码问题 …… 116
4.4 侦听器 …… 118
4.4.1 Servlet Context 侦听器 …… 118
4.4.2 ServletRequest 侦听器 …… 123
4.4.3 HttpSession 侦听器 …… 125
4.5 小结 …… 128
4.6 习题 …… 129
第 5 章 JavaBean 组件模型 …… 132
5.1 JavaBean 概述 …… 132
5.1.1 代码重用与组件规范 …… 132
5.1.2 JavaBean 的分类及特点 …… 132
5.1.3 JavaBean 的代码结构 …… 133
5.2 JavaBean 的编程 …… 134
5.2.1 JSP 动作与 JavaBean 生命周期 …… 134
5.2.2 封装业务逻辑的 JavaBean …… 140
5.3 一些有用的 JavaBean …… 142
5.3.1 数据封装与表单 JavaBean …… 142
5.3.2 文件上传和下载的 JavaBean 组件 …… 146
5.3.3 邮件发送的 JavaBean 组件 …… 154
5.3.4 用 POI 组件生成 Excel 报表 …… 160
5.4 小结 …… 161
5.5 习题 …… 162
第 6 章 JDBC 数据库访问技术 …… 163
6.1 SQL Server 2000 的安装 …… 163
6.1.1 SQL Server 2000 概述 …… 163
6.1.2 SQL Server 2000 个人版的安装 … 163
6.2 JDBC 概述 …… 166
6.2.1 ODBC 概述 …… 166
6.2.2 JDBC 概述 …… 167
6.2.3 JDBC 驱动程序的类型 …… 167
6.2.4 JDBC Type - 4 驱动程序的安装方法 …… 168
6.3 JDBC API 中关键的类和接口 …… 168
6.3.1 java.sql 包 …… 168
6.3.2 javax.sql 包 …… 168
6.3.3 常用的 JDBC API 类和接口 …… 168
6.4 JDBC 应用样例 …… 175
6.4.1 查询与分页样例 …… 175
6.4.2 预编译 SQL 语句 …… 182
6.4.3 存储过程的调用 …… 185
6.4.4 添加新记录 …… 188
6.4.5 删除记录 …… 190
6.4.6 修改记录 …… 192
6.4.7 JDBC 事务处理 …… 195
6.4.8 数据库与 Excel 报表的动态生成 …… 200
6.5 连接池技术 …… 204
6.6 小结 …… 209
6.7 习题 …… 210
第 7 章 EL 与 JSTL 标记库 …… 212
7.1 EL 表达式语言 …… 212
7.1.1 EL 与 EL 隐含对象 …… 212
7.1.2 在 EL 中访问 JSP 隐含对象的 getXXX() 方法 …… 217
7.1.3 用 EL 访问 JavaBean 中的属性 …… 218
7.2 JSTL 1.1 …… 218
7.2.1 JSTL 简介 …… 218

7.2.2 JSTL1.1 的安装 …… 219
7.2.3 JSTL 标记的结构和使用 …… 220
7.3 JSTL 核心标记库 …… 220
7.3.1 <c:set>标记 …… 220
7.3.2 <c:remove>标记 …… 223
7.3.3 <c:out>标记 …… 223
7.3.4 <c:catch>标记 …… 224
7.3.5 <c:if>标记 …… 224
7.3.6 <c:choose>标记 …… 225
7.3.7 <c:forEach>标记 …… 226
7.3.8 <c:forTokens>标记 …… 228
7.3.9 <c:import>标记 …… 228
7.3.10 <c:redirect>标记 …… 229
7.4 消息国际化 …… 229
7.5 SimpleTagSupport 与自定义标记 …… 233
7.5.1 自定义标记 …… 233
7.5.2 SimpleTagSupport 类 …… 233
7.5.3 SimpleTagSupport 的生命周期 …… 234
7.5.4 SimpleTagSupport 的标记库描述符 …… 234
7.6 Tag File 与自定义标记 …… 240
7.6.1 标记文件 …… 240
7.6.2 标记文件中常用的编程标记 …… 242
7.6.3 JSP 页面和标记文件间传递参数的方法 …… 247
7.7 小结 …… 249
7.8 习题 …… 250
第 8 章 Struts 1.3 框架 …… 251
8.1 JSP 开发模式 …… 251
8.1.1 Model1 开发模式 …… 251
8.1.2 MVC 设计模式与 Model2 开发模式 …… 251
8.2 Struts 1.3 框架 …… 256
8.2.1 Struts 1.3.8 的安装和配置 …… 256
8.2.2 struts-config.xml 部署文件 …… 257
8.3 Struts 的表单技术 …… 261
8.3.1 Struts 的表单标记 …… 261
8.3.2 ActionForm 表单类和表单校验 …… 262
8.3.3 DynaActionForm 表单类和 Validator 校验框架 …… 264
8.3.4 基于 Map-backed 的 ActionForm 表单类 …… 268
8.4 Struts 的控制器层 …… 271
8.4.1 Action 类与单一操作控制逻辑 …… 272
8.4.2 多重操作控制逻辑 …… 276
8.5 小结 …… 276
8.6 习题 …… 277
第 9 章 基于 Struts 1.3 的网上书店项目 …… 278
9.1 DAO 设计模式和简单工厂设计模式 …… 278
9.1.1 DAO 设计模式 …… 278
9.1.2 简单工厂设计模式与单实例设计模式 …… 278
9.2 网上书店系统功能分析 …… 279
9.3 数据库设计 …… 280
9.4 系统实现 …… 282
9.4.1 网站规划 …… 282
9.4.2 系统整体设计方案 …… 283
9.4.3 用户登录模块的实现 …… 286
9.4.4 用户注册模块的实现 …… 291
9.4.5 图书快速搜索模块的实现 …… 295
9.4.6 购物车模块的实现 …… 301
9.4.7 订单生成模块的实现 …… 307
9.5 小结 …… 310
9.6 习题 …… 311
参考文献 …… 312

第1章　JSP概述

本章要点：

- 静态网页与动态网页的本质区别以及各自的优缺点。
- Servlet 实现动态网页的基本原理。
- JSP 页面最终被编译为 Servlet 程序来运行。
- JDK 1.5、DreamWeaver 8、JC4 和 Tomcat 5.5 的安装与配置要点，Web 应用的概念与典型的文件夹结构，Tomcat 的文件夹结构，Tomcat 中 server.xml 配置文件的基本用法。
- 在 DreamWeaver 8 环境下 JSP 网页的编写与测试方法。

1.1　静态网页和动态网页

近年来，Internet 得到了快速的发展和普及，网络带宽基本上能够满足企业和个人的需要。Internet上常见的服务有：远程登录、文件传输、电子邮件、WWW(World Wide Web，简称为Web)信息浏览和信息搜索等。WWW 利用超文本(HyperText)技术把 Internet 中不同主机上的相关信息有机地组织在一起，用户可以通过浏览器访问这些信息。在搜索引擎的帮助下，用户能够快速地在 Web 中找到与某个主题相关的信息。

一个 Web 站点主要包含网页、数据库、程序等资源，站点中的信息一般是通过网页的形式向外界展示的，根据网页内容的存在形式，可以把网页分成两大类：静态网页和动态网页。

1.1.1　静态网页

1. 什么是静态网页

静态网页是指用 HTML 标记语言等来编排，页面中的内容固定不变，存盘后一般以 *.html、*.htm 等文件形式存在的网页。如果要更改网页的内容，需要网站维护人员重新编辑网页并上传到服务器中。静态网页中可以加入 Flash 动画、JavaScript 和 DHTML 等技术来获得一定的动态视觉效果，但网页中的关键内容还是以静态文本的形式存在，仍然属于静态网页范畴。

2. 静态网页的优点

静态网页的主要优点如下：

- 访问响应速度快。静态网页存储在服务器上，客户端通过浏览器发出资源访问请求，服务器响应后，将资源通过网络传给客户端浏览器，服务器不会运行静态网页中的任何代码，客户端能在比较短的时间内取得整个网页的内容。
- 容易被搜索引擎收录。搜索引擎把站点中的静态网页文件下载回来后，自动从网页中抽取有用的文本信息建立索引，有助于网站的推广。

3. 静态网页的缺点

静态网页的主要缺点如下：

- 缺乏交互性。现在网站中的很多重要信息是用数据库存储的，由于静态网页中的代码一般不会被服务器执行，所以网页无法从数据库中获取信息，使得静态网页缺乏交互性和灵活性。
- 维护工作量大。需要修改静态网页中的内容时，维护人员必须手工编辑静态网页。

为了克服静态网页的不足，人们引入了动态网页技术。

4. URI 与 URL

Web 上的资源(HTML 文档、图像、视频片段、程序等)通过一个通用资源标识符(Universal Resource Identifier，URI)进行定位。URI 一般由三部分组成：访问资源的命名机制、存放资源的主机名、资源的路径和文件名。

URL 是 Uniform Resource Locator 的缩写，译为“统一资源定位符”。URL 是 Internet 上用来描述信息资源的字符串，主要用在各种 WWW 客户程序和服务器程序上。采用 URL 可以用一种统一的格式来描述网络中的各种信息资源，包括文件、服务器的地址和目录等。URL 的格式由三部分组成：第一部分是协议(或称为服务方式)，第二部分是存有该资源的主机 IP 地址或域名(包括端口号)，第三部分是资源的具体地址。例如，在 http://localhost/jsp/exam.jsp 中，协议为 http，主机名为 localhost(默认端口号为 80)，资源地址为主机上的“/jsp/exam.jsp”。再如 http://localhost/exam.jsp? ID = 908&username = tomcat，“?”号后的 name = value 对称为 URL 查询串，表示客户端给 exam.jsp提交的参数，参数间用“&”符号连接，在此例中给 exam.jsp 传入了两个参数：ID 和 username。

URL 是 URI 命名机制的一个子集。

1.1.2 动态网页

1. 什么是动态网页

动态网页是指网页中的关键内容在服务器端动态生成的网页。例如，在网站中查找书名中包含有“JSP”关键字的图书时，用户首先把关键字通过网页表单提交给服务器端的动态网页，服务器运行动态网页中的程序代码，访问数据库，查询结果与其他静态网页内容合并后传回客户端浏览器解码显示。

动态网页和静态网页相比，最本质的区别在于：动态网页会被服务器当作程序来执行，对网页中的静态内容(如 HTML 标记)服务器不作任何处理，直接输出给客户端，而动态网页部分的代码会被服务器识别并执行；静态网页则不会被服务器视为程序，网页中的内容(包括程序代码)不会被服务器运行。

DHTML 是动态 HTML(Dynamic HTML，DHTML)，它是在 HTML 页面中加入 JavaScript 脚本程序，使其能根据用户的动作作出响应，例如，鼠标指针移动到图片上时，图片改变颜色。但是 DHTML 不是我们所说的动态网页，因为它的 JavaScript 脚本不是在服务器端执行，而是在客户端执行。

2. 动态网页的优点

动态网页的主要优点如下：

- 能够访问服务器端的数据库。动态网页具有数据库访问能力，大大增强了它的应用能力和适用范围。
- 具有交互性。动态网页根据用户传入的参数在服务器后台执行业务逻辑，最后将运行结果返回给用户，具有与用户交互的能力。
- 网页维护的工作量有所减少。由于许多数据可以存储在数据库中，利用动态网页的编程能力可动态地生成 HTML 内容，减少了网站维护人员的工作量。

3. 动态网页的缺点

动态网页的主要缺点如下：

- 不利于搜索引擎的信息收集。搜索引擎一般只能访问 Web 服务器中的文件，无法访问服务器中的数据库，存储在数据库中的信息无法被搜索引擎收集。

- 数据库访问是一个瓶颈。Web 服务器和数据库服务器之间是通过网络协议进行通信的，并发访问量大时会降低系统的响应性能。

1.2 Servlet 与 JSP

1.2.1 Servlet 技术

1. 什么是 Servlet 技术

Servlet 称为 Java 的服务器端应用小程序，是 Sun 公司的服务器端组件技术之一。Servlet 的基本功能与 CGI 类似，属于 Web 服务器扩展，是 Java 平台下实现动态网页的基本技术，具有占用资源少、效率高、可移植性和安全性强等特点。

Servlet 程序在 Servlet 容器中运行，嵌入了 Servlet 容器的 Web 服务器就具备提供 Servlet 服务的能力。一般 Web 服务器主要处理客户端对静态资源(如 *.htm 文件等)的请求，如果客户端请求的是 Servlet 资源，则 Web 服务器把这个请求转发给 Servlet 容器处理。Servlet 容器接收到客户端请求后，运行指定的 Servlet 程序，结果以 HTML 等形式返回给客户端。Servlet 容器作为一种插件嵌套在 Web 服务器中，通过扩展 Web 服务器的功能来提供 Servlet 服务。

典型的 Servlet 应用模型如图 1-1 所示。客户端通过 HTTP 协议请求运行在服务器端的 Servlet 程序，Servlet 程序从 HTTP 输入流中读取初始参数，运行业务逻辑代码完成计算，必要时可调用 EJB、JavaBean 或 JDBC，运行结果通过 Servlet 输出流返回给客户端。从图 1-1 中可以看出，Servlet 起到联系前后台的桥梁作用。

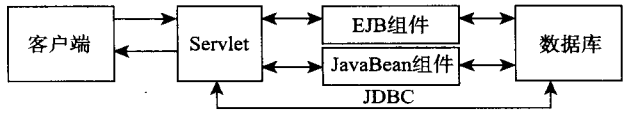

图 1-1 典型的 Servlet 应用模型

2. Servlet 2.4 的新特征

Servlet 2.4 相对 Servlet 2.3 变化不是很大，主要有以下几个方面的改变：

1)web.xml 的文件类型声明改用 XML Schema。web.xml 是 JSP 应用中重要的部署描述文件，部署信息的书写要遵照指定的格式，Servlet 2.3 是使用 DTD 文档来描述 web.xml 中部署信息的语法格式，Servlet 2.4 改用 XML Schema 文档来描述，语法定义更细、更灵活，并提供了更好的可扩充性。

2)新的 SevletRequest 接口。在 Servlet 2.4 中，ServletRequest 接口和它的实现类 ServletRequestWrapper 中新增了四个新的方法：getRemotePort()、getLocalName()、getLocalAddr()和 getLocalPort()。这四个新增的方法配合原有的其他方法，可以查询底层的 IP 连接细节。

3)RequestDispatcher 的变化。在早期版本的 Servlet 规范中，RequestDispatcher 和 Filters 过滤器相互作用时，对过滤器的激活条件和激活顺序等没有作出明确的定义，在 Servlet 2.4 规范中，提供了一个新的部署描述元素 <dispatcher> 来指明过滤器的作用时机和激活顺序。

4)Servlet Request 侦听器。在早期版本的 Servlet 规范中，只有 context 侦听器和 session 侦听器，在 Servlet 2.4 中新增了 ServletRequest 等侦听器。

5)SingleThreadModel 被取消。SingleThreadModel 是一个标记接口，没有定义任何方法。如果一个 Servlet 使用这个接口，Servlet 容器在运行这个 Servlet 程序的多个子线程时，一次只允许一个子线程的 service()方法被执行。SingleThreadModel 并没有解决多线程访问共享资源时的同步问题，

因而被取消。

6)其他更新如下:

- Servlet 2.4 只支持 HTTP 1.1 客户端。Servlet 2.3 既支持 HTTP 1.0 客户端,也支持 HTTP 1.1 客户端。
- 可用 Servlet 作 welcome 页面。而在 Servlet 2.3 中,能作 welcome 页面的只能是 HTML 文件或 JSP 文件。
- 支持 response 的国际化。在 Servlet 2.3 中,不能直接在 response 中定义返回给客户浏览器的字符编码。在 Servlet 2.4 中,可以通过 javax.servlet.ServletResponse.setCharacterEncoding() 方法或在 web.xml 中使用 <locale-encoding-mapping-list> 元素定义返回给客户端的信息编码标准。

1.2.2 JSP 概述

1. JSP 与 Servlet 的关系

Servlet 向客户端返回的内容需要用 out.print()输出,不便于网页版面的设计和修改。为了解决这个问题,Sun 公司制定了 JSP 技术规范,JSP 网页版面的设计与维护可通过 DreamWeaver 等工具软件来实现,比 Servlet 要直观和容易。

JSP 页面最终会被 JSP 服务器编译成一个 Servlet 程序来运行。

2. JSP 2.0 新特性

JSP 2.0 需要 Servlet 2.4 支持。JSP 2.0 中新增的特性主要有:

- 支持 EL 和 JSTL 1.1。EL(Expression Language,表达式语言)是一种相对独立的语言,主要用于 JSP 页面直接访问对象,语法简单。JSTL(JSP Standard Tag Library,JSP 标准标记库)提供一组类似于 HTML 的标记,方便 JSP 页面组织程序流程、读取 XML 文件和访问数据库等。使用 EL 表达式和标记库技术,能够大幅减少 JSP 页面中的 Java 代码,使 Java 程序员与网页设计人员的分离成为可能。
- 支持 SimpleTag 接口和 JSP Fragment,用户自定义标记更为简单。

1.3 上机实验环境的搭建

本书样例在调试时,使用的操作系统为 Windows XP Professional SP2,也可以在 Windows 2000 SP4 下运行,在安装软件之前,建议以 Windows 的 Administrator(管理员)身份登录操作系统。

1.3.1 安装 J2SE 1.5

J2SE(Java 2 Standard Edition, Java 2 标准版)主要为 PC 机和服务器提供 Java 的编程环境和运行时环境,一般也简称为 JDK(Java Development Kits, Java 开发包)。建议安装 Sun 公司的 J2SE 1.5 或以上版本,安装过程的注意事项如下。

1. J2SE 的安装文件夹

本书要求将 JDK 安装至 c:\jdk 文件夹中,以方便后续的配置。

2. J2SE 的运行参数配置

JDK 安装完毕后,需要为 JDK 配置三个系统环境变量:

- java_home = c:\jdk
- classpath = .; c:\jdk\lib\dt.jar; c:\jdk\lib\tools.jar
- path = ...; c:\jdk\bin

java _ home 环境变量指明 JDK 安装在哪个文件夹下，此变量在本书中的取值是“c:\jdk”。

classpath 变量指明编译 Java 程序时类库的搜索路径，此变量在本书中的取值为“.;c:\jdk\lib\dt. jar; c:\jdk\lib\tools. jar”。注意第一个搜索路径为“.”，表示优先在当前文件夹下搜索相关的支持类。tools. jar 和 dt. jar 是 JDK 中最常用的库文件。

path 指明 JDK 可执行文件所在的文件夹。Windows 系统中一般已经存在 path 变量，当前变量的取值不妨用“.....”(省略号)表示，现在只需要在 path 当前值的尾部附加上“; c:\jdk\bin”。

JDK 安装过程的视频请参考电子资源包 demo\ch1\jdk. exe 文件。

1.3.2 安装 Tomcat 5.5

1. Tomcat 5.5 的安装与配置

Tomcat 是支持 Servlet 和 JSP 技术规范的 Web 服务器软件，是 Apache 软件基金会管理下的一个开源项目，可免费下载使用，其核心代码来自 Sun 公司。本书使用 Tomcat 5.5.17，软件包名为 apache-tomcat-5.5.17.zip，用 WinRAR 3 解压文件，释放目标路径为“c:\”，最后要求将文件夹“c:\apache-tomcat-5.5.17”改名为“c:\tomcat”，以方便后续的配置。

最后为 Tomcat 添加环境变量：tomcat = c:\tomcat。

Tomcat 5.5 安装过程的视频请参考电子资源包 demo\ch1\tomcat. exe 文件。

2. Tomcat 5.5.17 的文件夹结构

安装完 Tomcat 5.5 后，本书获得的 Tomcat 5.5.17 的主要文件夹结构如图 1-2 所示。

在图 1-2 中，各文件夹的基本功能如下。

(1)c:\tomcat\bin 文件夹

这个文件夹主要存放 Tomcat 的脚本文件和可执行文件。

startup. bat 是 Windows 环境下启动 Tomcat 服务器的批处理文件，双击运行这个文件会弹出一个命令行窗口，显示启动过程的各步信息，如果关闭此窗口，则表示关闭 Tomcat 服务器。Tomcat 启动成功后，打开 IE5 浏览器，在地址栏中输入“http://127.0.0.1:8080”并回车后，会看到如图 1-3 所示的画面。建议把此文件发送到 Windows 桌面上，以方便启动 Tomcat。

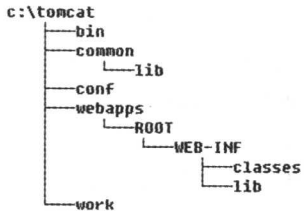

图 1-2 Tomcat 5.5.17 的主要文件夹结构

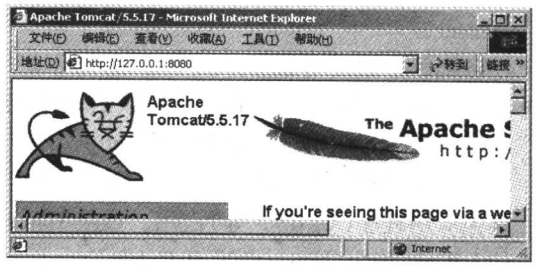

图 1-3 Tomcat 默认主站点的首页

(2)c:\tomcat\common\lib 文件夹

这个文件夹下存放 *.jar 形式的公用类库文件，这些类库文件可供 Tomcat 管理下的各个 Web 应用所共享。servlet-api. jar 文件是支持 Servlet 2.4 规范的类库文件，在编译 Servlet 程序时会用到这个库文件。数据库的 JDBC 驱动程序 *.jar 一般也放在此文件夹下。

(3)c:\tomcat\conf 文件夹

这个文件夹主要存放 Tomcat 的全局配置文件。

Tomcat 是一个基于组件的 Web 服务器，各组件在 server. xml 中定义和配置，如侦听端口号、虚拟主机、上下文(Context)、连接超时时间等。server. xml 的基本结构如下：

```
<Server>
  <Service>
    <Connector/>
    <Engine>
      <Host>
        <Context>
          <Resource/>
            <ResourceParams>
              <parameter/>
            </ResourceParams>
        </Context>
      </Host>
    </Engine>
  </Service>
</Server>
```

server. xml 文件中各元素的主要功能说明如下：

1）Server 组件：Server 是单实例（Singleton）的，它一般用来代表整个 JVM，在 JVM 中包含有一个或多个 Service 实例。Server 在指定端口上侦听“shutdown”关机命令。Server 不是容器，所以不要给它直接定义日志等组件。

2）Service 组件：一个 Service 中可以有多个 Connector 和一个 Engine。Engine 被 Service 中的一个或多个 Connector 所共享。Service 也不是容器。

3）Connector 组件：它是客户端和 Tomcat 容器类元素间的通信接口，用于接收客户端的 request 请求，然后转发给 Engine 处理，并把返回的处理结果传递给客户端。Tomcat 中此项的默认配置参数如下：

```
<Connector port = "8080" maxHttpHeaderSize = "8192"
                maxThreads = "150" minSpareThreads = "25" maxSpareThreads = "75"
                enableLookups = "false" redirectPort = "8443" acceptCount = "100"
                connectionTimeout = "20000" disableUploadTimeout = "true"
                URIEncoding = "GB2312"/>
                ......
</Connector>
```

上述各配置参数说明如下：

- port 是侦听端口号，即 Tomcat 默认的 HTTP 连接端口号为 8080，客户端访问 Tomcat 的默认虚拟主机时，URL 的一般写法为“http://127. 0. 0. 1:8080”。HTTP 协议默认的端口号是 80，在此也可以把“8080”改为“80”，前提是“80”端口在当前 Windows 系统中是空闲的，没有被其他应用程序（如 IIS 5 等）占用。如果 Tomcat 使用了 80 端口，则访问 Tomcat 默认虚拟主机的 URL 写法为“http://127. 0. 0. 1”。
- maxThreads = "150" ：定义最多能使用 150 个线程来处理来自客户端的连接请求。
- minSpareThreads = "25" ：定义保留的最少空闲线程数。
- maxSpareThreads = "75"：定义最大空闲线程数，例如，某时刻使用了 100 个线程来处理客户端的请求，这些线程运行完毕后最多保留 75 个空闲线程，另外的 25 个线程将被撤销。
- acceptCount = "100"：当线程数达到 maxThreads 时，如果还有新的连接请求到来，则把这些请求放到队列中等候调度，此参数定义队列中最多容纳多少个等候请求，这个限额之外的连接请求到来时将会被拒绝，并给客户端返回“Connection refused”的错误信息。
- connectionTimeout = "20000"：定义请求在等候队列中的最大等候时间，单位为毫秒，等待超时后 Tomcat 会给客户端返回“等待超时”的错误信息。

- enableLookups = "false"：关闭 DNS 查询功能，在并发访问流量较大时，能够节约系统开销和网络开销。在实际应用中，DNS 一般由专门的 DNS 主机来完成，Tomcat 不必承担这部分工作。
- redirectPort = "8443"：当客户端使用 https 安全协议访问虚拟主机时，Connector 将此请求转发给 8433 端口上的 Connector 处理。
- URIEncoding = "GB2312"：此参数告知服务器 URL 中的字符编码标准为 GB2312。

4）Engine 组件：Engine 是处理 request 请求的入口。当把 Tomcat 当作独立的 Web 服务器时，Engine 实例为 Tomcat 提供分析 HTTP 报头的功能，从中抽取出 request 请求，并转发给合适的虚拟主机。Tomcat 此项的默认配置是：

```
<Engine name = "Catalina" defaultHost = "localhost">
  ……
</Engine>
```

上述配置参数的意思为：如果收到形如 http://localhost:8080 或 http://127.0.0.1:8080 的请求，则 Engine 将请求转发给 Tomcat 中名为 localhost 的主机处理。

5）Host 组件：一个 Host 代表一台虚拟主机。所谓虚拟主机，就是把一台运行在互联网上的 Web 服务器划分成多个“虚拟”的服务器，每一个虚拟主机有独立的域名，各虚拟主机独立工作。一个Host 的典型配置如下：

```
<Host name = "www. my1. com" appBase = "c:\host2"
                    unpackWARs = "true" autoDeploy = "true">
        ……
</Host>
```

上述配置参数说明如下：

- name = "www. my1. com"：定义虚拟主机的域名为 www. my1. com。
- appBase = "c:\host2"：此项参数定义虚拟主机默认的 Web 应用发布文件夹，在此文件夹下建立的下级文件夹会被视为一个 Web 应用的“根”文件夹，文件夹名也被视为 Web 应用的上下文路径名。例如，如果存在 c:\host2\k1\sport. htm 资源，则“k1”文件夹会被视为一个上下文路径名为“k1”的 Web 应用，客户端通过“http://www. my1. com: 8080/k1/sport. htm”访问 Web 应用中的 sport. htm 资源。在 Tomcat 中，默认虚拟主机 localhost 默认的 Web 应用发布文件夹为“c:\tomcat\webapps”，将一个 Web 应用的文件夹或 *.war 文件复制到此文件夹下，则启动 Tomcat 时 Tomcat 自动发布此 Web 应用，Web 应用的上下文路径名就是 Web 应用的文件夹名。
- unpackWARs = "true"：此项参数取值为 true 时，Tomcat 会把以 *.war 形式发布到虚拟主机中的 Web 应用自动解压缩，并存放在 c:\host2 中；此项参数取值为 false 时，表示不解压缩。
- autoDeploy = "true"：此项参数取值为 true 时，表示启动 Tomcat 后，自动发布虚拟主机中的各个 Web 应用。

Tomcat 默认的虚拟主机 localhost 定义如下：

```
<Host name = "localhost" appBase = "webapps"
            unpackWARs = "true" autoDeploy = "true"
                  xmlValidation = "false" xmlNamespaceAware = "false">
        ……
</Host>
```

在上述配置中，默认虚拟主机的域名为"localhost"，由 Windows XP 在本地解释，在 c:\WINDOWS\system32\drivers\etc\host 文件中，定义 localhost 的 DNS 解释为“127. 0. 0. 1 localhost”，即

域名 localhost 的 IP 地址是 127.0.0.1(本机)。

6)Context 组件：一个 Context 代表虚拟主机中的一个 Web 应用。

Web 应用是一个有相对独立功能的 Web 子系统，它相当于 DreamWeaver 中“站点”的概念。JSP 的 Web 应用主要由 Servlet 程序、JSP 文件、HTML 文件、Java 类以及其他的资源构成。一个 Web 应用存储在某个文件夹下，或用 jar 打包为 *.wart 文件。每个 Web 应用有相似的文件夹结构，其中的 WEB-INF 等文件夹有特定的用途，在后续的章节中会介绍。多个 Web 应用虽然可同属一个虚拟主机，但相互间有一定的独立性，例如，运行时不共享彼此的 *.jar 类库等，这种相对独立性方便各 Web 应用间并行开发、独立调试和独立迁移。

为了方便客户端通过 URL 访问 Web 应用中的资源，需要给每个 Web 应用定义一个 URL 名，即虚拟路径，在 JSP 中称之为上下文路径(Context Path)。在 URL 中通过上下文路径指明访问的是虚拟主机中哪个 Web 应用中的资源，此时的 URL 写法为：

```
http://主机名：端口/上下文路径名/资源路径/资源名
```

上下文路径起到“根”目录的作用，例如，“/bbs”上下文路径表示一个论坛 Web 应用的根目录，如果 Web 应用根目录下有一个名为“exam.jsp”的资源，则此资源的 URI 可写成“/bbs/exam.jsp”，再加上访问协议和主机名后就得到此资源的 URL。

可用 <Context> 元素定义一个 Web 应用的上下文路径，它的典型使用格式如下：

```
<Host name = "www.my1.com" appBase = "c:\host2"
      unpackWARs = "true" autoDeploy = "true">
         <Context docBase = "my1" path = "" reloadable = "true"/>
         <Context docBase = "d:\data" path = "/news" reloadable = "true"/>
</Host>
```

在上述配置信息中，<Host> 元素定义了一个名为 www.my1.com 的虚拟主机，其默认的 Web 应用发布文件夹为 c:\host2。

<Context> 元素中常用的参数说明如下：

- docBase：指明 Web 应用所在的物理文件夹名。如果 Web 应用存储在默认的发布文件夹下，直接写出 Web 应用的文件夹名(相对路径)即可，如果 Web 应用存储在其他文件夹下，一般要求给出 Web 应用文件夹的绝对物理路径。
- path：定义 Web 应用的上下文路径名。上下文路径名的第 1 个字符一般为“/”。如果写为 path = ""，双引号内没有任何字符，则表示这是虚拟主机的默认 Web 应用，它的上下文路径名为“/”。
- reloadable：此属性取值为 true 时，表示让 Catalina 监控 Web 应用中 WEB-INF\classes 文件夹和 WEB-INF\lib 文件夹下文件的变化，如果发现这些文件夹下的类或配置文件发生了变化，则自动重载此 Web 应用。这个特性在开发 Web 应用期间有一定的用处，但它占用相当大的运行时系统开销，在正式发布 Web 应用后，建议取消此项特性。Tomcat 是周期性地检测文件的变化情况的，当类文件发生变化时，Tomcat 可能不会立即发现并作出重载反应，因此在开发过程中，更新一个类后，需要用 Tomcat 中 manager 角色的用户来重载 Web 应用，重启 Tomcat 也可以重载 Web 应用。
- cookies：取值为 true 时表示使用 cookies 存储 session 隐含对象的 ID 号。如果取值为 false，则使用 URL 重写技术存储 session 隐含对象的 ID 号。默认值为 true。
- backgroundProcessorDelay：此项属性取值为一个正整数 n 时，表示让 Web 应用在 n 秒钟延时后启动一个后台处理线程检查 session 超时或 WEB-INF\classes 文件夹下程序文件的变化。默认取值为 -1，表示使用父主机中的处理线程完成检查任务。

- crossContext：取值为 true 时，表示允许当前 Web 应用通过 ServletContext. getContext( )方法获取当前虚拟主机中其他 Web 应用的 request 转发对象(RequestDispatcher)。默认值为 false，表示调用 getContext( )方法时会返回 null。

在上述样例中，<Context docBase = "my1" path = "" reloadable = "true"/> 指明：有一个 Web 应用存放在 c:\host2\my1 文件夹下，其上下文路径名为“/”，即此 Web 应用是虚拟主机的默认 Web 应用，通过 http://www. my1. com:8080 来访问它的默认首页。

<Context docBase = "d:\data" path = "/news" reloadable = "true"/> 指明：此虚拟主机中还有另外一个 Web 应用，其物理文件夹为“d:\data”。path = "/news" 为此 Web 应用的上下文路径名，在 http://www. my1. com:8080/news/2006/a1. htm 中，“/news”是 Web 应用的上下文路径名，它充当虚拟“根”目录的作用，“/2006”表示虚拟“根”目录下有一个名为“2006”的一级文件夹，资源 a1. htm 存储在此文件夹中，即此资源在服务器上的绝对路径为：d:\data\2006\a1. htm。

在一台虚拟主机中，一个 Web 应用的上下文路径名应该是唯一的。

Tomcat 采用 URL 的最长可能前缀与每个 Web 应用的上下文路径相匹配，从而确定一个 URL 请求应该交由哪个 Web 应用处理。例如，在上例的两个 <Context> 中，如果存在两个不同的资源：c:\host2\my1\news\a2. htm 和 d:\data\a2. htm，则访问它们的 URL 均写为 http://www. my1. com:8080/news/a2. htm。第一个资源的 Web 应用上下文路径名为“/”，第二个资源的 Web 应用上下文路径名为“/news”，所以 Tomcat 接收到此 URL 请求后，交由上下文路径名为“/news”的 Web 应用处理，而不是默认的 Web 应用“/”。

在 Tomcat 的六大组件中，Server、Service 和 Connector 不是容器类组件，不能直接处理客户端的请求，也不能直接生成响应。

(4)c:\tomcat\work 文件夹

这是一个临时文件夹，主要存放各 JSP 文件编译后得到的 Servlet 程序、日志和临时文件等。

(5)c:\tomcat\webapps 文件夹

Tomcat 默认的虚拟主机是 localhost，它默认的 Web 应用发布文件夹是 webapps。在此文件夹下发布的 Web 应用通过 URL 即可直接访问，不需要在 <Context> 中进行配置。

webapps\ROOT 文件夹是 localhost 主机默认的 Web 应用，其主页通过 http://127. 0. 0. 1:8080 来访问。webapps\tomcat-doc 中存储的是 Tomcat 5. 5 的说明文档。webapps\jsp-examples 下存储了一些 JSP 样例。webapps\servlets-examples 下存放有一些 Servlet 样例。

如果需要给 localhost 虚拟主机创建一个 Web 应用，最简单的方法是在 webapps 下创建一个文件夹，在此文件夹下组织 Web 应用。

一个典型的 Web 应用文件夹结构如图 1-2 中的 ROOT 文件夹所示。ROOT 为 Web 应用的文件夹名，在 ROOT 下可以创建下级文件夹来分类存放资源文件。WEB-INF 是一个特殊的文件夹，主要存放当前 Web 应用的配置文件、类库文件等资源。这个文件夹对于客户端而言是不可访问的，但对 Web 应用中的程序而言是可访问的。例如，把 index. jsp 网页文件复制到此文件夹后，客户端通过 http://127. 0. 0. 1:8080/WEB-INF/index. jsp 无法成功访问，因此，如果希望某些网页资源不被客户端直接访问或下载，可以把它们存储在此文件夹下，在服务器端设计 JSP/Servlet 程序来间接访问它们。需要注意的是，WEB-INF 文件夹不一定能阻止搜索引擎的访问。WEB-INF\lib下主要存放当前 Web 应用中用到的 *.jar 类库文件。WEB-INF\classes 下主要存放 *.class 形式的类文件。WEB-INF\web. xml 文件是当前 Web 应用的部署描述文件，例如，Servlet 程序的部署信息就写在此文件中。本书默认使用 webapps\ROOT 作为学习和上机环境，相关的资源文件就存储在此文件夹下，并通过 http://127. 0. 0. 1:8080 来访问。

1.3.3 安装 DreamWeaver 8 简体中文版

1. DreamWeaver 8 的安装

JSP 网页文件中的 HTML 标记一般不需要手工书写，可利用网页设计工具 DreamWeaver 8(简写为 DW8)来自动生成。DreamWeaver 8 简体中文版可从国内的 http://www.onlinedown.net 或官方的 http://www.adobe.com/products/dreamweaver/ 网站中下载。双击运行 DreamWeaver 安装程序，各安装步骤取默认值，直至安装结束。

本书使用 Tomcat 默认虚拟主机 localhost 中默认的 Web 应用(ROOT)作为学习环境，需要把“c:\tomcat\webapps\ROOT”文件夹定义成 DreamWeaver 8 的一个 JSP 站点，访问站点的 URL 为：http://127.0.0.1:8080。

2. DreamWeaver 8 的工作界面

DreamWeaver 8 的工作界面如图 1-4所示，简要说明如下：

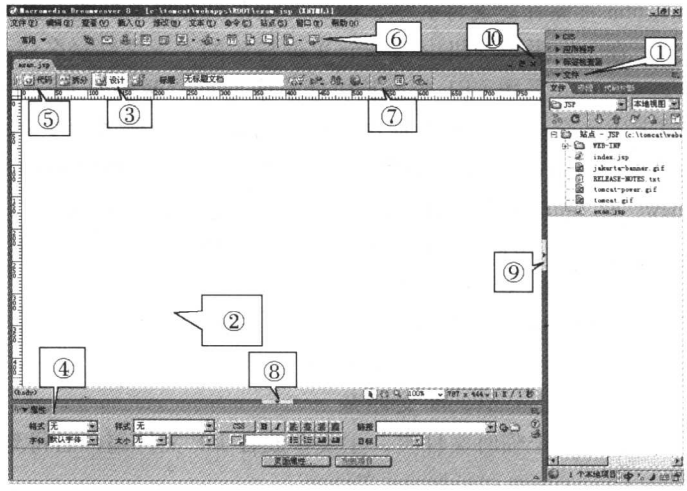

图 1-4 DreamWeaver 8 工作界面

①号位置为面板区，图中当前展开的是【文件】窗口，窗口中列出当前站点中的文件。如果要新建一个 JSP 文件，在【文件】窗口中适当的位置上单击鼠标右键，在弹出的右键菜单中选择【新建文件】，则自动生成一个文件名类似“untitledxx.jsp”的 JSP 文件，将文件的基本名修改为所期望的名字即可。要编辑此文件，可用鼠标双击文件名。

②号位置为文档窗口。③、⑤、⑦所在的位置称为文档工具栏。③号位置上的按钮是【设计视图】按钮，在设计视图方式下，用户可在②号位置的编辑区域中以“所见即所得”的方式设计和编辑 JSP 页面，在④号位置的【属性】面板中设定所选中的网页元素的属性。

⑥号位置为插入栏，它包含用于创建和插入页面对象的按钮，共有 9 个类别，即常用、布局、表单、文本、HTML、服务器代码、应用程序、Flash 元素和收藏夹，利用插入栏上的按钮可快速地在页面中插入表格、表单等网页元素。

⑤号位置的按钮为【代码视图】按钮，在代码视图方式下，②号位置的文档窗口中显示的是当前 JSP 文件中的 HTML 标记和代码，此状态下可编辑 JSP 代码。

⑦号位置的图标按钮用于预览当前 JSP 网页，单击此图标，在弹出的下拉列表中选择【预览在 iexplore】，会激活 IE 浏览器预览当前编辑的 JSP 页面。

⑧号位置的下三角小按钮用于隐藏和显示【属性】面板，⑨号位置的右三角小按钮用于显示和隐藏右边的面板窗口区。

⑩号位置上的“×”按钮用于关闭当前 JSP 网页文件。

DW8 的安装、配置 JSP 站点、生成第一个 JSP 页面的操作视频请参考电子资源包 demo\ch1\dw8.exe 文件。