第 10 章

动态网页语言——JSP 基础语法

学习指引

本章将介绍动态网页语言——JSP 的基础、运行机制、页面基本结构、JSP 注释方法以及 page 指令的使用，还有一些练习及就业面试技巧。其实 JSP 的核心语法都是根据 Java 演变而来的，比如 Java 中的各种判断、循环语句在 JSP 中都可以使用，接下来我们就一起来学习一下 JSP 的基础语法。

重点导读

- 掌握 JSP 的运行机制。
- 掌握 JSP 中注释语句的使用。
- 了解并学会使用 JSP 基本语法。
- 了解 JSP 页面的基本结构。
- 学会使用 JSP 注释。
- 了解并掌握 page 指令的使用方法。

10.1 JSP 简介

JSP 全名为 Java Server Pages，即 Java 服务器页面，是一个简化的 Servlet 设计，它是由 Sun Microsystems 公司倡导、许多公司参与一起建立的一种动态网页技术标准。JSP 技术有点儿类似 ASP 技术，它是在传统的网页 HTML 文件中插入 Java 程序段和 JSP 标记，从而形成 JSP 文件，后缀名为.jsp。用 JSP 开发的 Web 应用是跨平台的，既能在 Linux 下运行，也能在其他操作系统上运行。

它实现了 HTML 语法中的 Java 扩展（以<%，%>形式）。JSP 与 Servlet 一样，是在服务器端执行的。通常返回给客户端的就是一个 HTML 文本，因此客户端只要有浏览器就能浏览。

JSP 技术使用 Java 编程语言编写类 XML 的 tags 和 scriptlets，来封装产生动态网页的处理逻辑。网页还能通过 tags 和 scriptlets 访问存在于服务端的资源的应用逻辑。JSP 将网页逻辑与网页设计的显示分离，支持可重用的基于组件的设计，使基于 Web 的应用程序的开发变得迅速和容易。JSP(Java Server Pages)是一种

动态页面技术，它的主要目的是将表示逻辑从 Servlet 中分离出来。

Java Servlet 是 JSP 的技术基础，而且大型的 Web 应用程序的开发需要 Java Servlet 和 JSP 配合才能完成。JSP 具备了 Java 技术的简单易用，完全面向对象，具有平台无关性且安全可靠，主要面向因特网的所有特点。

10.2　JSP 运行机制

JSP 机制概述：可以把 JSP 页面的执行分成两个阶段，一个是转译阶段，一个是请求阶段。

转译阶段：JSP 页面转换成 Servlet 类。

请求阶段：Servlet 类执行，将响应结果发送至客户端，可分为以下 6 步完成。

（1）用户（客户机）访问响应的 JSP 页面，如 http://localhost:8080/test/hello.jsp。

（2）服务器找到相应的 JSP 页面。

（3）服务器将 JSP 转译成 Servlet 的源代码。

（4）服务器将 Servlet 源代码编译为 class 文件。

（5）服务器将 class 文件加载到内存并执行。

（6）服务器将 class 文件执行后生成 HTML 代码发送给客户机，客户机浏览器根据相应的 HTML 代码进行显示。

如果该 JSP 页面为第一次执行，那么会经过这两个阶段，而如果不是第一次执行，那么将只会执行请求阶段。这也是为什么第二次执行 JSP 页面时明显比第一次执行要快的原因。

如果修改了 JSP 页面，那么服务器将发现该修改，并重新执行转译阶段和请求阶段。这也是修改页面后访问速度变慢的原因。JSP 运行机制关系图如图 10-1 所示。

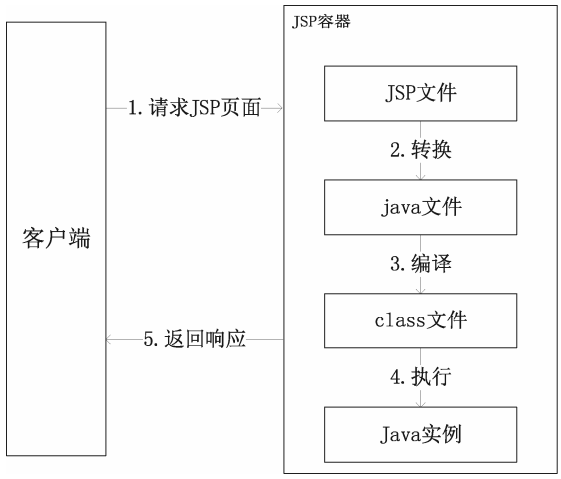

图 10-1　JSP 运行机制关系图

下面分别介绍一下 JSP 引擎、Web 容器和 Servlet 容器。

1. JSP 引擎

JSP 引擎实际上要把 JSP 标签、JSP 页中的 Java 代码甚至连同静态 HTML 内容都转换为大块的 Java 代

码。这些代码块被 JSP 引擎组织到用户看不到的 Java Servlet 中去，然后 Servlet 自动把 JVM（Java 虚拟机）编译成 Java 字节码。这样，当网站的访问者请求一个 JSP 页时，在他不知道的情况下，一个已经生成的、预编译过的 Servlet 实际上将完成所有的工作，非常隐蔽而又高效。因为 Servlet 是编译过的，所以网页中的 JSP 代码不需要在每次请求该页时被解释一遍。JSP 引擎只需在 Servlet 代码最后被修改后编译一次，然后这个编译过的 Servlet 就可以被执行了。由于是 JSP 引擎自动生成并编译 Servlet，不用程序员动手编译代码，所以 JSP 能带来高效的性能和快速开发所需的灵活性。

2. Web 容器和 Servlet 容器

Servlet 容器的主要任务是管理 Servlet 的生命周期。Web 容器更准确地说应该叫 Web 服务器，它是用来管理和部署 Web 应用的。还有一种服务器叫作应用服务器，它的功能比 Web 服务器要强大得多，因为它可以部署 EJB 应用，可以实现容器管理的事务，一般的应用服务器有 WebLogic 和 WebSphere 等，它们都是商业服务器，功能强大但都是收费的。Web 容器最典型的就是 Tomcat 了，Tomcat 是 Web 容器也是 Servlet 容器。

10.3 JSP 页面的基本结构

JSP 页面通常由 5 种元素组合而成：普通的 HTML 标记符，JSP 标记（如指令标记、动作标记），成员变量和方法，Java 程序片和 Java 表达式。

【例 10-1】JSP 程序基本结构说明。

```
<%@ page contentType="text/html;charset=GBK" %>
<!-- jsp 指令标签  定义当前页面的属性 -->
<!-- contentType 希望客户解析器来解析执行的方式    -->
<!-- charset=ISO-8859-1 字节编码,GBK 亚洲文字双字节符,gb2312 中国汉字编码规范-->
<%@ page import="Java.util.Date"  %>
<!-- JSP 指令标签 import 引入 Java 的核心类 -->
<!-- 申明 Java 成员变量和方法,多个客户共享 -->
<%!   Date date;                          //数据声明
      int sum;
      public int add(int m,int n)         //方法声明
      {   return m+n;
      }
  %>
<html>
<body bgcolor=cyan> <font size=4> <!--  html 标记 -->
<!-- 插入 Java 程序片: -->
   <% Date date=new Date();               //创建 Date 对象
      //这里的变量是局部变量
      out.println("<br>"+date);
      sum=add(12,34);
   %>
  <BR>在下一行输出和:<br>
   <%= sum+100 %>
   <!-- Java 表达式,必须能求值,服务器计算,以字符形式显示在客户端-->
</font>
</body>
</html>
```

以上对 JSP 页面的每一个组成部分都进行了非常详细的注释，运行结果如图 10-2 所示。

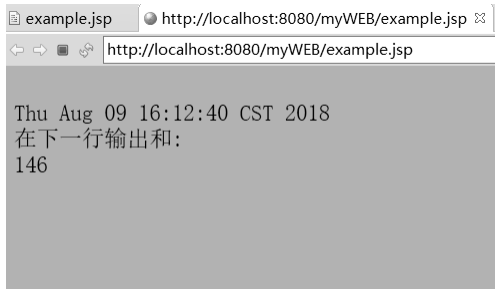

图 10-2　example.jsp 运行结果

10.4　JSP 注释

JSP 网页中通过在 HTML 中嵌入 Java 脚本语言来响应页面的动态请求，即其 JSP 代码中包含 HTML 标记和 Java 脚本。

1. HTML 注释

JSP 页面使用这种注释且客户端通过浏览器查看 JSP 源文件时，能够看到 HTML 注释文字，其语法格式是：

```
<!--要注释的文字-->
```

【例 10-2】使用 HTML 注释。

```
<%@ page language="Java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<title></title>
</head>
<body>
Hello World!<br/>
<!-- HTML 注释,该部分注释在网页中不会被显示-->
</body>
</html>
```

运行上面的 JSP 文件，页面显示“Hello World!”且没有显示 HTML 注释，如图 10-3 所示。

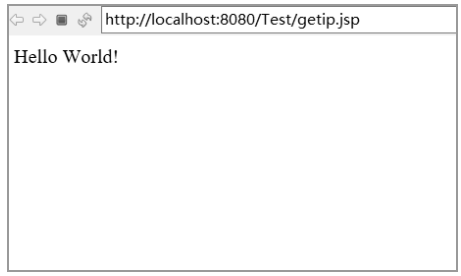

图 10-3　HTML 注释

右击页面，选择“查看源文件”，就可以看到这条 HTML 注释，如图 10-4 所示。

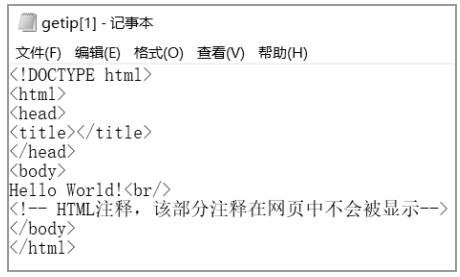

图 10-4　查看源

2. JSP 注释

使用这种注释时，JSP 引擎编译该页面时会忽略 JSP 注释。JSP 注释主要有两个作用：为代码做注释以及将某段代码注释掉。下面是其语法格式。

```
<%--要注释的文字--%>
```

【例 10-3】使用 JSP 注释。

```
<%@ page language="Java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>JSP 注释</title>
</head>
<body>
<%-- 该部分注释在网页中不会被显示--%>
<p>
   今天的日期是: <%= (new Java.util.Date()).toLocaleString()%>
</p>
</body>
</html>
```

上面的程序通过使用 JSP 注释<%--该部分注释在网页中不会被显示--%>，页面上不会显示，查看源文件也不会显示，运行结果如图 10-5 所示。

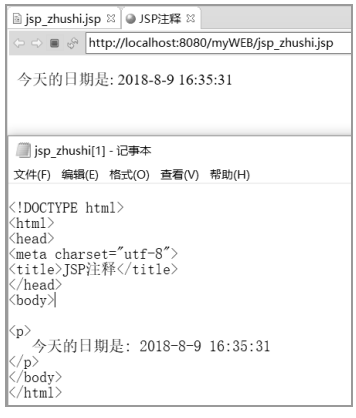

图 10-5　使用 JSP 注释

3. JSP 脚本中的注释

所谓的脚本就是嵌入<%和%>标记之间的程序代码，使用的语言是 Java，因此在脚本中进行注释和在 Java 类中进行注释的方法一样，其格式如下。

```
<% //单行注释%>;
<% /*多行注释*/ %>。
```

Java 中提供的多行注释，客户端也是无法看见的。

10.5　page 指令

page 指令是在 JSP 开发中较为重要的，使用此属性，可以定义一个 JSP 页面的相关属性，包括设置 MIME 类型、定义需要导入的包、错误页的指定等。

page 指令语法：

```
<%@ page 属性="内容"%>
```

page 指令的主要属性如表 10-1 所示。

表 10-1　page 指令属性表

序号	指 令 属 性	描　　述
1	autoFlush	可以设置为 true 或 false，如果设置为 true，当缓冲区满时，到客户端的输出被刷新；如果设置为 false，当缓冲区满时，将出现异常，表示缓冲区溢出。默认为 true，例如：autoFlash="true"
2	buffer	指定到客户端输出流的缓冲模式。如果为 none，则表示不设置缓冲区；如果指定数值，那么输出的时候就必须使用不小于这个值的缓冲区进行缓冲。此属性要和 autoFlush 一起使用。默认不小于 8KB，根据不同的服务器可以设置
3	contentType	定义 JSP 字符的编码和页面响应的 MIME 类型，如果是中文的话则使用如下形式：contentType="text/html;charset=GBK"
4	errorPage	定义此页面出错时要跳转的显示页，例如：errorPage="error.jsp"，要与 isErrorPage 属性一起使用
5	extends	主要定义此 JSP 页面产生的 Servlet 是从哪个父类扩展而来，例如：extends="父类名称"
6	import	此 JSP 页面要导入哪几个操作包，例如：import="Java.util.*"
7	info	此 JSP 页面的信息，例如：info="text info"
8	isErrorPage	可以设置 true 或 false，表示此页面是否为出错的处理页，如果设置成 true，则 errorPage 指定的页面出错时才能跳转到此页面进行错误处理；如果设置成 false，则无法处理
9	isThreadSafe	可以设置 true 或 false，表示此页面是否是线程安全的，如果为 true，表示一个 JSP 页面可以处理多个用户的请求；如果为 false，则此 JSP 一次只能处理一个用户请求
10	language	用来定义要使用的脚本语言，目前只能是“Java”，例如：language="Java"
11	pageEncoding	JSP 页面的字符编码，默认值为 pageEncoding="iso-8859-1"，如果是中文则可以设置为 pageEncoding="GBK"
12	session	可以设置 true 或 false，指定所在页面是否参与 HTTP 会话。默认值为 true，例如：session="true"

【例 10-4】使用 page 指令设置文件编码。

```
<%@ page language="Java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<html>
```

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
<h2>这是我的个人主页</h2>
</body>
</html>
```

本页面使用 pageEncoding 属性将整个页面的编码设置为 UTF-8，运行结果如图 10-6 所示。

图 10-6　页面显示

10.6　综合案例

了解了 JSP 的运行机制、注释和基础的语法，现在来编写一个 JSP 文件，程序使用 if-else 判断语句编写。

【例 10-5】判断语句 if-else 的使用。

```
<%@ page language="Java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>IF...Else 实例</title>
</head>
<body>
    <%!int day = 7;%>
    <%
        if (day == 6 → day == 7) {
    %>
    <p>今天周末,不用上班, ~</p>
    <%
        } else {
    %>
    <p>今天工作日,需要上班 ~</p>
    <%
        }
    %>
</body>
</html>
```

以上程序使用了 if-else 语句，day 为 1～7 为判断条件，来判断是否需要上班，若 day 等于 1～5 中任意一个值则需要上班，等于 6 或 7 则不用上班，运行结果如图 10-7 所示。

图 10-7　if-else 语句使用

中文编码问题：在 JSP 网页中也可能出现中文乱码问题，那是因为中文编码不同引起的，如果要在页面中正常显示中文，需要在 JSP 文件头部添加以下代码用于设置网页中文编码方式为 UTF-8。

```
<%@ page language="Java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
```

【例 10-6】获取 IP 地址并用中文显示。

```
<%@ page language="Java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>中文编码问题</title>
</head>
<body>
Hello World!<br/>
<%
out.println("你的 IP 地址 " + request.getRemoteAddr());
%>
</body>
</html>
```

运行以上 JSP 文件，输出 IP 地址为 127.0.0.1，如图 10-8 所示。

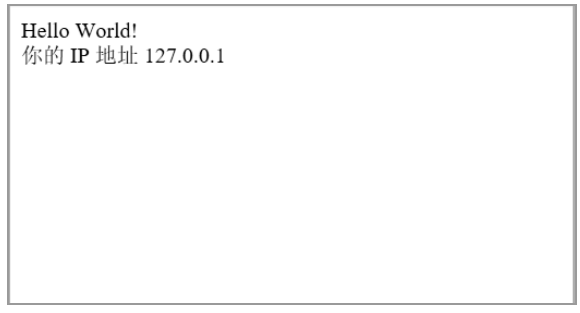

图 10-8　获取 IP 地址

JSP 基础语法还包括 JSP 声明、JSP 表达式、JSP 指令、JSP 动作以及 JSP 内置对象，这些内容在后续章节详细说明。

10.7 就业面试解析与技巧

10.7.1 面试解析与技巧（一）

面试官：什么是 JSP?

应聘者：Java Server Pages 是一项支持动态内容的 Web 网页开发技术，它可以帮助开发人员通过利用特殊的 JSP 标签在 HTML 页面中插入 Java 代码，这些标签大部分都是以<%和%>为标记。

面试官：使用 JSP 的优点是什么？

应聘者：JSP 提供了如下所列的几个优势：因为 JSP 允许在 HTML 页面本身中嵌入动态元素而不是使用一个单独的 CGI 文件，所以性能明显更好一点儿；JSP 在被服务器处理之前总会被编译一次，它不像 CGI/Perl 那样，每一次页面被请求时，服务器都需要加载一个解释器和目标脚本；Java Server Pages 是在 Java Servlets API 之上创建的，所以 JSP 可以像 Servlets 那样也可以访问所有强大的企业级的 Java APIs，包括 JDBC、JNDI、EJB、JAXP 等；JSP 页面可以结合 Servlets 进行使用，处理业务逻辑，该模型是通过 Java Servlet 模板引擎支持的。

10.7.2 面试解析与技巧（二）

面试官：和单纯的 Servlet 相比，JSP 的优势是什么？

应聘者：与通过使用大量的 println 语句生成的 HTML 相比，JSP 在编写（和修改）常规的 HTML 上更方便。其他的优点是：在 HTML 页面中嵌入 Java 代码，跨平台，创建数据库驱动的 Web 应用程序，服务器端的编程功能。

面试官：和 JavaScript 相比，JSP 的优势是什么？

应聘者：JavaScript 可以在客户端生成动态的 HTML，却不能和 Web 服务器进行交互来完成复杂的任务，例如，数据库的访问和图像处理等。