第 3 章 JSP语法

学习目的与要求

本章主要介绍 JSP 脚本元素、JSP 指令标记和 JSP 动作标记。通过本章的学习，要求读者理解 JSP 页面的组成部分，掌握 JSP 语法，并能够使用 JSP 开发 Web 页面。

本章主要内容

- JSP 页面的基本构成
- JSP 脚本元素
- JSP 指令标记
- JSP 动作标记

一个 JSP 页面通常由 HTML 标记、JSP 注释、Java 脚本元素以及 JSP 标记 4 种基本元素组成。这 4 种基本元素在 JSP 页面中是如何被使用的为本章介绍的重点。

本章涉及的 JSP 页面保存在 ch3 项目的 src/main/webapp 目录中。

扫一扫

视频讲解

3.1 JSP 页面的基本构成

▶ 3.1.1 一个 JSP 页面

在 HTML 静态页面文件中加入和 Java 相关的动态元素，就构成了一个 JSP 页面。一个 JSP 页面通常由以下 4 种基本元素组成：

(1) 普通的 HTML 标记。

(2) JSP 注释。

(3) Java 脚本元素，包括声明、Java 程序片和 Java 表达式。

(4) JSP 标记，例如指令标记、动作标记和自定义标记等。

【例 3-1】 根据 example3_1.jsp 代码中的注释识别 JSP 页面的基本元素。

example3_1.jsp 的代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<jsp:include page="a.jsp"/><!-- JSP 动作标记 -->
<%!
int i = 0;                    //数据声明
int add(int x, int y) {       //方法声明
      return x + y;
}
 %>
<html>       <!-- HTML 标记 -->
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
```

```
</head>
<body>
   <%
   i ++;                          //Java 程序片
   int result = add(1, 2);
    %>
   i 的值为<% = i %><% -- Java 表达式 -- %>
   <br>
   1 + 2 的和为<% = result %>
</body>
</html>
```

a.jsp 的代码如下：

```
<% @ page language = "java" contentType = "text/html; charset = UTF - 8" pageEncoding = "UTF - 8" %>
<! DOCTYPE html >
< html >
< head >
< meta charset = "UTF - 8">
< title > Insert title here </title >
</head >
< body >
     被 example3_1.jsp 动态引用.
</body >
</html >
```

▶ 3.1.2 JSP 页面注释

在 example3_1.jsp 的代码中有许多 JSP 注释，添加注释能够增强 JSP 文件的可读性，便于 Web 项目的更新和维护。JSP 页面中常见的注释有以下两种：

❶ HTML 注释

格式：<!--HTML 注释-->

在标记符“<!--”和“-->”之间加入注释内容，就构成了 HTML 注释。

JSP 引擎对于 HTML 注释也要进行处理，即不将它看作注释，如果其中有 JSP 代码，也将被 JSP 引擎处理。JSP 引擎将处理之后的 HTML 注释交给客户端，在通过浏览器查看 JSP 源文件时能够看到 HTML 注释。

❷ JSP 注释

格式：<%--JSP 注释--%>

在标记符“<%--”和“--%>”之间加入注释内容，就构成了 JSP 注释。

JSP 引擎将 JSP 注释当作真正的注释，在编译 JSP 页面时忽略这部分代码，因此在通过浏览器查看 JSP 源文件时无法看到 JSP 注释。

▶ 3.1.3 实践环节——识别 JSP 页面元素

识别出以下 JSP 页面的基本元素：

```
<% @ page language = "java" contentType = "text/html; charset = UTF - 8" pageEncoding = "UTF - 8" %>
<! DOCTYPE html >
<!-- 学习 JSP 页面的基本构成 -->
<% !
```

```
        String content = "JSP 页面的基本构成：";
%>
<html>
<head>
<meta charset = "UTF - 8">
<title> shijian3_1.jsp </title>
</head>
<body>
<%
     content = content + "HTML 标记、JSP 注释、JSP 标记以及 Java 脚本元素";
%>
<% = content %>
</body>
</html>
```

扫一扫

视频讲解

3.2 JSP 脚本元素

JSP 中的 Java 脚本元素包括声明、Java 程序片以及 Java 表达式。

▶ 3.2.1 Java 程序片

在标记符“<%”和“%>”之间插入的 Java 代码被称为 JSP 页面的 Java 程序片。Java 程序片的格式如下：

```
<% Java 代码 %>
```

一个 JSP 页面中可以有任意段 Java 程序片，这些程序片将被 JSP 引擎（本书中指 Tomcat 服务器）按顺序执行。在一个程序片中声明的变量称为 JSP 页面的局部变量，它们在 JSP 页面中后继的所有程序片以及表达式内都有效。

当多个用户请求一个 JSP 页面时，JSP 引擎为每个用户启动一个线程，不同的线程会分别执行该 JSP 页面中的 Java 程序片，程序片中的局部变量会在不同的线程中被分配不同的内存空间，因此一个用户对 JSP 页面中局部变量操作的结果不会影响其他用户。Java 程序片的执行原理如图 3.1 所示。

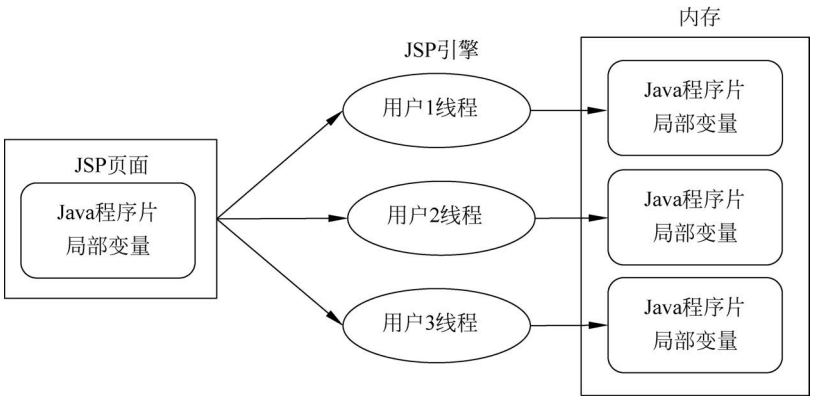


图 3.1 Java 程序片的执行原理

【例 3-2】 编写一个 JSP 页面 example3_2.jsp，在页面中存在一段 Java 程序片，该程序片内声明了一个整型的局部变量 n，初始值为 0。

example3_2.jsp 的代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>example3_2.jsp</title>
</head>
<body>
<%
    int n = 0;
    n++;
    out.print("n = " + n);
%>
</body>
</html>
```

如果有 5 个用户请求 example3_2.jsp 页面，JSP 引擎会启动 5 个线程，页面中的 Java 程序片在每个线程中都会被执行一次，共执行 5 次；在内存中，局部变量 n 对应 5 处不同的存储空间，初始值都为 0，且都仅执行了一次自加运算，所以 5 个用户看到的页面效果是相同的。

3.2.2 成员变量与方法的声明

成员变量和方法的声明格式如下：

```
<%! 变量或方法的定义 %>
```

在标记符“<%!”和“%>”之间声明的变量被称为 JSP 页面的成员变量，它们可以是 Java 语言允许的任何数据类型。例如：

```
<%!
    int n = 0;
    Date date;
%>
```

成员变量在整个 JSP 页面内都有效(与书写位置无关)，因为 JSP 引擎在将 JSP 页面转译成 Java 文件时将这些变量作为类的成员变量，这些变量的内存空间直到服务器关闭才释放，因此多个用户共享 JSP 页面的成员变量，任何用户对 JSP 页面成员变量操作的结果都会影响到其他用户。

在标记符“<%!”和“%>”之间声明的方法被称为 JSP 页面的成员方法，该方法在整个 JSP 页面内有效，但是在该方法内定义的变量仅在该方法内有效。

【例 3-3】 编写一个 JSP 页面 example3_3.jsp，在该页面中声明一个成员变量 n(初始值为 0)和方法 add()(求两个整数的和)，另外该页面中还有一段 Java 程序片，在程序片中声明一个局部变量 m，并且对成员变量 n 和局部变量 m 分别进行自加运算。

example3_3.jsp 的代码如下：

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

```
<html>
<head>
<meta charset="UTF-8">
<title>example3_3.jsp</title>
</head>
<%!
int n=0;
int add(int x,int y){
    return x+y;
}
%>
<body>
<%
    int m=0;
    n++;
    m++;
    int result=add(1,2);
    out.print("成员变量 n 的值为:"+n+"<br>");
    out.print("局部变量 m 的值为:"+m+"<br>");
    out.print("1+2="+result+"<br>"+"<br>");
    out.print("第"+n+"个用户");
%>
</body>
</html>
```

在 example3_3.jsp 中，变量 n 是成员变量，被所有用户共享；变量 m 是局部变量，被每个用户独享。如果有 3 个用户请求这个 JSP 页面，则看到的效果如图 3.2 所示。

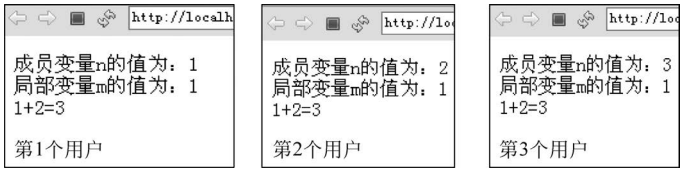


图 3.2　3 个用户请求 example3_3.jsp 页面的效果

从 example3_3.jsp 中可知 Java 程序片具有以下特点：

（1）调用 JSP 页面声明的方法。

（2）操作 JSP 页面声明的成员变量。

（3）声明局部变量。

（4）操作局部变量。

▶ 3.2.3　Java 表达式

在标记符“<%=”和“%>”之间可以插入一个表达式，这个表达式必须能求值。表达式的值由 Web 服务器负责计算，并将计算结果用字符串形式发送到客户端，作为 HTML 页面的内容显示。

在 Java 表达式中可以有算术表达式、逻辑表达式、条件表达式等，但在使用 Java 表达式时需要注意以下两点：

（1）不可以在“<%=”和“%>”之间插入语句，即输入的内容不能以分号结束。

（2）“<%=”是一个完整的符号，在“<%”和“=”之间不能有空格。

3.2.4 实践环节——在 JSP 页面中输出英文字母表

编写一个 JSP 页面，在 JSP 页面中使用 Java 程序片输出小写的英文字母表。

3.2.5 实践环节——网站访问量的统计

利用成员变量被所有用户共享这一性质实现一个简单的计数器，页面效果如图 3.3 所示。

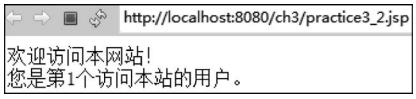
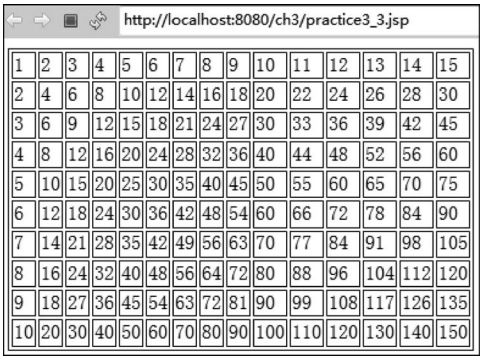


图 3.3 简单的计数器

3.2.6 实践环节——打印表格

在浏览器中输出 15×10 的表格，页面效果如图 3.4 所示。



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	4	6	8	10	12	14	16	18	20	22	24	26	28	30
3	6	9	12	15	18	21	24	27	30	33	36	39	42	45
4	8	12	16	20	24	28	32	36	40	44	48	52	56	60
5	10	15	20	25	30	35	40	45	50	55	60	65	70	75
6	12	18	24	30	36	42	48	54	60	66	72	78	84	90
7	14	21	28	35	42	49	56	63	70	77	84	91	98	105
8	16	24	32	40	48	56	64	72	80	88	96	104	112	120
9	18	27	36	45	54	63	72	81	90	99	108	117	126	135
10	20	30	40	50	60	70	80	90	100	110	120	130	140	150

图 3.4 15×10 的表格

扫一扫
视频讲解

3.3 JSP 指令标记

常用的 JSP 指令标记有 page 指令标记和 include 指令标记。

3.3.1 page 指令标记

page 指令标记用来定义整个 JSP 页面中的一些属性和这些属性的值，可以用一个 page 指令标记指定多个属性的值，也可以使用多个 page 指令标记分别为每个属性指定值。page 指令标记的语法格式如下：

```
<%@ page 属性 1 = "属性 1 的值" 属性 2 = "属性 2 的值" … %>
```

或

```
<%@ page 属性 1 = "属性 1 的值" %>
<%@ page 属性 2 = "属性 2 的值" %>
<%@ page 属性 3 = "属性 3 的值" %>
…
<%@ page 属性 n = "属性 n 的值" %>
```

page 指令标记的属性主要有 contentType、import、language 和 pageEncoding 等。

❶ contentType 属性

JSP 页面使用 page 指令标记只能为 contentType 属性指定一个值，用来确定响应的 MIME 类型（MIME 类型就是设定某种文件用相应的一种应用程序来打开的方式类型）。当用户请求一个 JSP 页面时，服务器会告诉浏览器使用 contentType 属性指定的 MIME 类型来解释执行所接收到的服务器为之响应的信息。例如，当浏览器使用 Word 应用程序打开用户的请求时，可以将 contentType 属性值设置为：

```
<%@page contentType = "application/msword;charset = UTF - 8" %>
```

常见的 MIME 类型有 text/html（HTML 解析器，所谓的网页形式）、text/plain（普通文本）、application/pdf（PDF 文档）、application/msword（Word 应用程序）、image/jpeg（JPEG 图像）、image/png（PNG 图像）、image/gif（GIF 图形）以及 application/vnd. ms-powerpoint（PowerPoint 应用程序）。

❷ import 属性

在 JSP 页面中使用 page 指令标记可以为 import 属性指定多个值，import 属性的作用是为 JSP 页面引入包中的类，以便在 JSP 页面的程序片、变量及方法声明或表达式中使用包中的类。

❸ language 属性

language 属性用来指定 JSP 页面中使用的脚本语言，目前该属性的值只能取"java"。

❹ pageEncoding 属性

contentType 中的 charset 是指服务器发送给浏览器时用户所见到的网页内容的编码；pageEncoding 是指 JSP 文件存储时所用的编码。

在 JSP 规范中，如果 pageEncoding 属性存在，那么 JSP 页面的字符编码方式就由 pageEncoding 决定，否则由 contentType 属性中的 charset 决定，如果 charset 也不存在，JSP 页面的字符编码方式采用默认的 ISO-8859-1。

【例 3-4】 编写一个 JSP 页面 example3_4.jsp，当用户请求该页面时，在 Eclipse 内嵌的浏览器中启动本地的 PowerPoint 应用程序打开该页面。

example3_4.jsp 的代码如下：

```
<%@ page language = "java" contentType = "application/vnd.ms - powerpoint; charset = UTF - 8"
pageEncoding = "UTF - 8" %>
<!DOCTYPE html>
<html>
<head>
<meta charset = "UTF - 8">
<title>example3_4.jsp</title>
</head>
<body>
    在学习 page 指令标记时，请记住只能为 JSP 页面设置一个 contentType 属性值，可以为 import 属性设置多个值.
</body>
</html>
```

▶ 3.3.2 include 指令标记

一个网站中的多个 JSP 页面有时需要显示同样的信息，例如该网站的 Logo、导航条等，为了便于维护网站程序，通常在这些 JSP 页面的适当位置嵌入一个相同的文件。include 指令标

记的作用就是将 JSP 文件、HTML 网页文件或其他文本文件等静态嵌入当前的 JSP 网页中，该指令标记的语法格式如下：

```
<%@include file="文件的 URL"%>
```

所谓静态嵌入就是“先包含后处理”，在编译阶段完成对文件的嵌入，即先将当前 JSP 页面与要嵌入的文件合并成一个新的 JSP 页面，然后由 JSP 引擎将新页面转化为 Java 文件处理并运行。

【例 3-5】 编写两个 JSP 页面 example3_5.jsp 和 example3_5_1.jsp，在 example3_5.jsp 页面中使用 include 指令标记静态嵌入 example3_5_1.jsp 页面，访问 example3_5.jsp 页面，运行效果如图 3.5 所示。

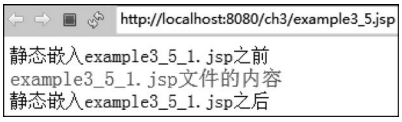


图 3.5 include 指令标记的使用

example3_5.jsp 的代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>example3_5.jsp</title>
</head>
<body>
    静态嵌入 example3_5_1.jsp 之前
    <br>
    <%@include file="example3_5_1.jsp"%>
    <br>
    静态嵌入 example3_5_1.jsp 之后
</body>
</html>
```

example3_5_1.jsp 的代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>example3_5_1.jsp</title>
</head>
<body>
    <font color="red" size=4>example3_5_1.jsp 文件的内容</font>
</body>
</html>
```

example3_5.jsp 页面静态嵌入 example3_5_1.jsp 页面，需要先将 example3_5_1.jsp 中的所有代码嵌入 example3_5.jsp 的指定位置，形成一个新的 JSP 文件，然后将新文件提交给 JSP 引擎处理，如图 3.6 所示。

在使用 include 指令标记时，需要注意嵌入文件后必须保证新合成的 JSP 页面符合 JSP 的语法规则，比如例 3-5 的 example3_5.jsp 和 example3_5_1.jsp 两个页面的 page 指令标记就不能指定不同的 contentType 值，否则合并后的 JSP 页面就使用两次 page 指令标记为 contentType 属性设置了不同的属性值，导致语法错误。

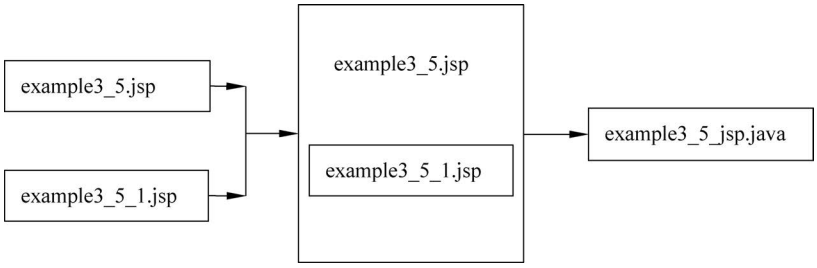


图 3.6　静态嵌入的原理

▶ 3.3.3　实践环节——制作导航栏

编写 3 个 JSP 页面 index.jsp、main.jsp 和 head.jsp，在 index.jsp 和 main.jsp 页面中分别使用 include 指令标记静态嵌入 head.jsp（导航栏）。导航栏的运行效果如图 3.7 所示。

图 3.7　导航栏的运行效果

扫一扫

视频讲解

3.4　JSP 动作标记

常用的 JSP 动作标记有 include、forward、param、useBean、getProperty 和 setProperty，其中 useBean、getProperty 和 setProperty 将在本书的第 5 章中介绍。

▶ 3.4.1　include 动作标记

动作标记 include 的作用是将 JSP 文件、HTML 网页文件或其他文本文件等动态嵌入当前的 JSP 网页中。该动作标记的语法格式如下：

```
<jsp:include page = "文件的 URL"/>
```

或

```
<jsp:include page = "文件的 URL">
      子标记
<jsp:include/>
```

当动作标记 include 不需要子标记时使用上述第一种形式。

所谓动态嵌入就是“先处理后包含”，在运行阶段完成对文件的嵌入，即在将 JSP 页面转译成 Java 文件时并不合并两个页面，而是在 Java 文件的字节码文件被加载并执行时才去处理 include 动作标记中引入的文件。与静态嵌入相比，动态嵌入的执行速度稍慢，但是灵活性较高。

【例 3-6】　编写两个 JSP 页面 example3_6.jsp 和 example3_6_1.jsp，在 example3_6.jsp 页面中使用 include 动作标记动态嵌入 example3_6_1.jsp 页面，运行 example3_6.jsp 页面。

example3_6.jsp 的代码如下：

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

```
<head>
<meta charset = "UTF - 8">
<title> example3_6.jsp </title>
</head>
<body>
    动态嵌入 example3_6_1.jsp 之前
    <br>
    <jsp:include page = "example3_6_1.jsp"/>
    <br>
    动态嵌入 example3_6_1.jsp 之后
</body>
</html>
```

example3_6_1.jsp 的代码如下：

```
<% @ page language = "java" contentType = "text/html; charset = UTF - 8" pageEncoding = "UTF - 8" %>
<! DOCTYPE html>
<html>
<head>
<meta charset = "UTF - 8">
<title> example3_6_1.jsp </title>
</head>
<body>
    <font color = "red" size = 4> example3_6_1.jsp 文件的内容</font>
</body>
</html>
```

页面文件 example3_6.jsp 通过动作标记 include 动态嵌入了 example3_6_1.jsp，此时 JSP 引擎不会将两个文件合并成一个 JSP 页面，而是分别将文件 example3_6.jsp 和 example3_6_1.jsp 转化成对应的 Java 文件和字节码文件。当 JSP 解释器解释执行 example3_6.jsp 页面时会遇到动作指令<jsp:include page＝"example3_6_1.jsp"/>对应的代码，此时才会执行 example3_6_1.jsp 页面对应的字节码文件，然后将执行的结果发送到客户端，并由客户端负责显示这些结果，所以 example3_6.jsp 和 example3_6_1.jsp 页面中 page 指令标记的 contentType 属性值可以不同。

▶ 3.4.2 forward 动作标记

动作标记 forward 的作用是从该标记出现处停止当前 JSP 页面的继续执行，转而执行 forward 动作标记中 page 属性值指定的 JSP 页面。该动作标记的语法格式如下：

```
<jsp: forward page = "文件的 URL"/>
```

或

```
<jsp: forward page = "文件的 URL">
      子标记
</jsp: forward>
```

当动作标记 forward 不需要子标记时使用上述第一种形式。

【例 3-7】 编写 3 个 JSP 页面 example3_7.jsp、oddNumber.jsp 和 evenNumbers.jsp，在 example3_7.jsp 页面中随机获取 0～10 的整数，当该整数为偶数时转向页面 evenNumbers.jsp，否则转向页面 oddNumber.jsp。首先访问 example3_7.jsp 页面。

example3_7.jsp 的代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>example3_7.jsp</title>
</head>
<body>
<%
long i = Math.round(Math.random() * 10);
if(i%2 == 0){
      System.out.println("获得的整数是偶数,即将跳转到偶数页面 evenNumbers.jsp.");
 %>
      <jsp:forward page="evenNumbers.jsp"/>
<%
      System.out.println("我是偶数,尝试一下能看到我吗?");
}else{
      System.out.println("获得的整数是奇数,即将跳转到奇数页面 oddNumber.jsp.");
 %>
<jsp:forward page="oddNumber.jsp"/>
<%
      System.out.println("我是奇数,尝试一下能看到我吗?");
}
 %>
</body>
</html>
```

evenNumbers.jsp 的代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>evenNumbers.jsp</title>
</head>
<body>
    我是偶数页.
</body>
</html>
```

oddNumber.jsp 的代码如下：

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>oddNumber.jsp</title>
</head>
<body>
    我是奇数页.
</body>
</html>
```

3.4.3 param 动作标记

动作标记 param 不能单独使用，但可以作为 include、forward 动作标记的子标记来使用，该动作标记以“名字-值”对的形式为对应页面传递参数。该动作标记的语法格式如下：

```
<jsp:父标记 page = "接收参数页面的 URL">
    <jsp:param name = "参数名" value = "参数值"/>
</jsp:父标记>
```

接收参数的页面可以使用内置对象 request 调用 getParameter("参数名")方法获取动作标记 param 传递过来的参数值。内置对象将在本书的第 4 章中介绍。

用户可以使用 param 子标记向页面传递多个参数，格式如下：

```
<jsp:父标记 page = "接收参数页面的 URL">
    <jsp:param name = "参数名 1" value = "参数值 1"/>
    <jsp:param name = "参数名 2" value = "参数值 2"/>
    <jsp:param name = "参数名 3" value = "参数值 3"/>
    …
</jsp:父标记>
```

【例 3-8】 编写两个页面 example3_8.jsp 和 computer.jsp，在页面 example3_8.jsp 中使用 include 动作标记动态包含文件 computer.jsp，并向它传递一个矩形的长和宽；computer.jsp 接收到参数后计算矩形的面积，并显示结果。

example3_8.jsp 的代码如下：

```
<%@ page language = "java" contentType = "text/html; charset = UTF - 8" pageEncoding = "UTF - 8" %>
<!DOCTYPE html>
<html>
<head>
<meta charset = "UTF - 8">
<title> example3_8.jsp </title>
</head>
<body>
    加载 computer.jsp 页面计算矩形的面积<br><br>
    <jsp:include page = "computer.jsp">
        <jsp:param value = "10" name = "length"/>
        <jsp:param value = "6" name = "width"/>
    </jsp:include>
</body>
</html>
```

computer.jsp 的代码如下：

```
<%@ page language = "java" contentType = "text/html; charset = UTF - 8" pageEncoding = "UTF - 8" %>
<!DOCTYPE html>
<html>
<head>
<meta charset = "UTF - 8">
<title> computer.jsp </title>
</head>
<body>
    <%
        String m = request.getParameter("length");
```

```
            String n = request.getParameter("width");
            double a = Double.parseDouble(m);
            double b = Double.parseDouble(n);
            out.print("我是被加载的页面,负责计算矩形的面积" + "<br>");
            out.print("给我传递的矩形的长度是:" + a + ",宽度是:" + b + "<br>");
            out.print("矩形的面积是:" + a * b);
        %>
</body>
</html>
```

▶ 3.4.4 实践环节——include 和 param 动作标记的应用

编写 3 个 JSP 页面 input.jsp、first.jsp 和 second.jsp，将 3 个 JSP 文件保存在同一个 Web 服务目录中，input.jsp 使用 include 动作标记加载 first.jsp 和 second.jsp 页面。first.jsp 页面可以画出一个表格，second.jsp 页面可以计算出两个正整数的最大公约数。当 first.jsp 被加载时获取 input.jsp 页面中 include 动作标记的 param 子标记提供的表格的行数和列数，当 second.jsp 被加载时获取 input.jsp 页面中 include 动作标记的 param 子标记提供的两个正整数的值。

▶ 3.4.5 实践环节——登录验证

编写 3 个 JSP 页面 login.jsp、validate.jsp 和 success.jsp，login.jsp 输入用户名和密码信息，提交给 validate.jsp 进行用户验证，如果验证为合法用户（用户名为 tom，密码为 jenny），则转到（forward 动作标记）success.jsp 页面，否则转到 login.jsp 页面重新登录。

本章小结

本章主要介绍了 JSP 页面的组成、JSP 脚本元素和 JSP 标记。一个 JSP 页面通常由 HTML 标记、JSP 注释、Java 脚本元素以及 JSP 标记组成。JSP 脚本元素包括 Java 程序片、JSP 页面成员变量与方法的声明、Java 表达式。JSP 标记包括指令标记和动作标记。

习 题 3

扫一扫

习题

扫一扫

自测题

第 4 章 JSP内置对象

学习目的与要求

本章主要介绍 request、response、out、session、application、pageContext、page、config 和 exception 等内置对象。通过本章的学习，要求读者理解 JSP 内置对象的含义，掌握 JSP 内置对象的使用方法。

本章主要内容

- request 对象
- response 对象
- out 对象
- session 对象
- application 对象
- pageContext 对象
- page 对象
- config 对象
- exception 对象

有些对象在 JSP 页面中不需要声明和实例化，可以直接在 Java 程序片和 Java 表达式部分使用，称这样的对象为 JSP 内置对象。JSP 内置对象由 Web 服务器负责实现和管理，JSP 自带了 9 个功能强大的内置对象，共分为 4 类。

(1) 与 Input/Output 有关的内置对象：与 Input/Output 有关的内置对象包括 request、response 和 out，该类对象主要用来作为客户端和服务器之间通信的桥梁。request 对象表示客户端对服务器发送的请求；response 对象表示服务器对客户端的响应；而 out 对象负责把处理结果输出到客户端。

(2) 与 Context 有关的内置对象：与 Context(上下文)有关的内置对象包括 session、application 和 pageContext。其中 session 对象表示浏览器与服务器会话的上下文环境；application 对象表示应用程序(Web 应用)的上下文环境；pageContext 对象表示当前 JSP 页面的上下文环境。

(3) 与 Servlet 有关的内置对象：与 Servlet 有关的内置对象包括 page 和 config。page 对象表示 JSP 文件转换为 Java 文件后的 Servlet 对象；config 对象表示 JSP 文件转换为 Java 文件后的 Servlet 的 ServletConfig 对象。

(4) 与 Error 有关的内置对象：与 Error 有关的内置对象只有一个 exception 对象。当 JSP 网页有错误时将产生异常，该对象用来处理这个异常。

本章涉及的 JSP 页面保存在 ch4 项目的 src/main/webapp 目录中。