第3章

JSP基本语法

JSP 页面可以嵌入 HTML 标签、指令、动作、脚本、扩展标签等，这些内容可以分成元素和模板数据。元素是在 JSP 基本语法中定义的内容，JSP 容器在转换阶段将元素翻译成相应的 Java 代码。JSP 页面中其他的所有内容都是模板数据。JSP 容器对模板数据不做处理。

JSP 定义的元素有 4 种类型：①指令，用于设置整个页面属性；②脚本，JSP 中嵌入的 Java 代码；③动作，利用 XML 语法格式的标记来控制 JSP 容器的行为；④表达式语言(EL)，$(表达式)，计算表达式括号内的表达式的值，将其转换为 String 类型并显示。


在线视频

3.1 JSP 页面的基本结构

在传统的 HTML 页面文件中加入 Java 程序片段和 JSP 标记就构成了一个 JSP 页面。一个 JSP 页面可由以下 5 种元素组合而成。

(1) 普通的 HTML 标记符。

(2) JSP 标记，如指令标记、动作标记。

(3) 变量和方法的声明。

(4) Java 程序片段。

(5)Java 表达式。

```
<% -- JSP 指令标记 -- %>
<%@ page import = "java.util.Date" %>
<%@ page language = "java" contentType = "text/html; charset = UTF - 8"
            pageEncoding = "UTF - 8" %>
<% -- 普通 HTML 标记 -- %>
<!DOCTYPE html>
<html>
<head>
<meta charset = "UTF - 8">
<title>JSP 元素组成</title>
</head>
<body>
```

```
<p>用户信息</p>
<% -- Java 程序片段 -- %>
<%
  String userid = (String) session.getAttribute("userID");
  UserDao userDao = new UserDao();
  Users users = userDao.getUsersByName(userid);
%><% -- 变量和方法声明 -- %>
<%!
  SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:ss:mm");
  String start_time = "2020-09-17 00:00:00";
  String end_time = "2022-06-31 00:00:00"
  Date d_start_time = df.parse(start_time);
  Date d_end_time = df.parse(end_time);
  public void sub(Date d_start_time, Date d_end_time) {
    System.out.println(d_end_time.getTime() - d_start_time.getTime());
  }
%>
<% -- Java 表达式 -- %>
<ul class="reg_ul">
<li>
<span>姓名:</span>
<span><% =users.getUsername() %></span>
</li>
<li>
<span style="margin-right: 20px;">邮箱:</span>
<span><% =users.getEmail() %></span>
</li>
</ul>
</body>
</html>
```

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,该线程负责执行常驻内存的字节码文件来响应相应用户的请求。这些线程由 Tomcat 服务器来管理,将 CPU 的使用权在各个线程之间快速切换,以保证每个线程都有机会执行字节码文件。

字节码文件的任务如下。

(1) JSP 页面中普通的 HTML 标记符号,交给客户的浏览器执行显示。

(2) JSP 标记、变量和方法声明、Java 程序片段由 Tomcat 服务器负责执行,将需要显示的结果发送给客户的浏览器。

(3) Java 表达式由 Tomcat 服务器负责计算,将结果转换为字符串,交给客户的浏览器负责显示。

3.2 变量和方法的声明

3.2.1 变量声明

在“<%!”和“%>”标记符之间声明变量,变量的类型可以是 Java 语言允许的任何数据结构类型,这些变量称为 JSP 页面的成员变量。

```
<%!
  Date date = new Date();
  String username = "Jack";
%>
```

“<%!”和“%>”之间声明的变量在整个 JSP 页面内都有效，与“<%!”“%>”标记符在 JSP 页面中所在的书写位置无关。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，这些线程由 JSP 引擎来管理，这些线程共享 JSP 页面的成员变量，因此任何一个用户对 JSP 页面成员变量操作的结果，都会影响到其他用户。共享变量的概念如图 3-1 所示。

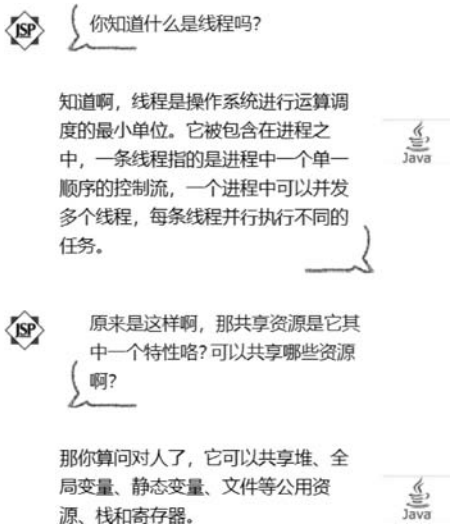

图 3-1　共享变量的概念

Example3_1_accumulative_counter.jsp 中通过具体事例对成员变量这一特性做详细的介绍，实现一个简单的累计计数器，比较成员变量和局部变量的区别。

Example3_1_accumulative_counter.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
        pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>累计计数器</title>
</head>
<body>
欢迎使用累计计数器,带你了解成员变量的概念!<br>
<%! int i_member=0; %>
<%  int i_local=0; %>
<%
out.print("刷新之后,成员变量的值为" + this.i_member++ + ",成员变量 i_member 的值发生改变<br>");
out.print("刷新之后,局部变量的值为" + i_local++ + ",局部变量的 i_local 的值不变");
%>
</body>
</html>
```

第一次加载该案例时，成员变量和局部变量均为初始值 0，实现效果如图 3-2 所示。

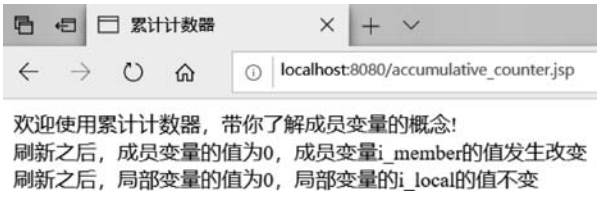

图 3-2 初次加载效果

每次刷新当前页面，成员变量的值将发生改变，执行+1 操作，而局部变量的值保持不变，刷新 3 次后实现效果如图 3-3 所示。

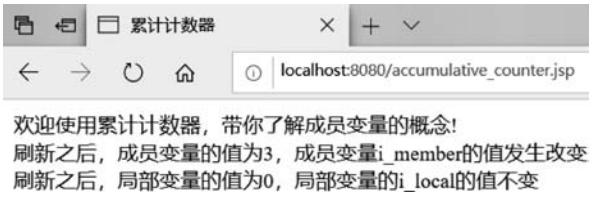

图 3-3 累计计数器刷新 3 次实现效果

3.2.2 方法声明

在“<%!”和“%>”标记符号之间定义方法，所定义的方法在整个 JSP 页面有效，可以在 Java 程序片段中被调用。

```
<%!
  public long getDate() throws ParseException {
    SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:ss:mm");
    String time = "2020-09-17 00:00:00";
    Date d_time = df.parse(time);
    return d_time.getTime();
  }
%>
```

方法内声明的变量只在该方法内有效，当方法被调用时，方法内声明的变量被分配内存，方法被调用完毕即可释放这些变量所占的内存。变量存储概念解释如图 3-4 所示。

Example3_2_method_statement.jsp 中通过具体事例对方法声明做了详细的介绍和了解，在“<%!”和“%>”之间定义了 add()和 sub()两个方法，分别为自增和自减，然后在程序片段中调用这两个方法。

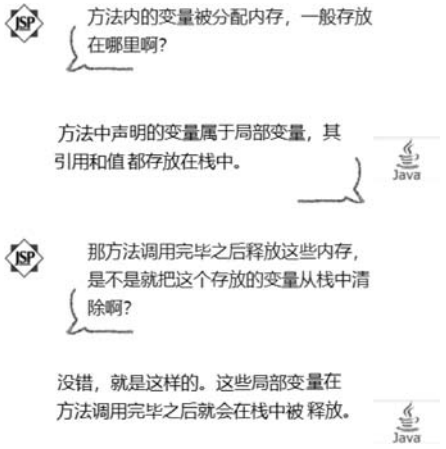

图 3-4 变量存储概念

Example3_2_method_statement.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
        pageEncoding="UTF-8" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>方法声明之自增自减</title>
</head>
<body>
<%!
    int add(int i) {
        return ++i;
    }

    int sub(int i) {
        return --i;
    }
%>
<% int i=0;
    out.print("变量"+i+"的原值为:"+i+",调用自增方法计算变量" + i + "的之后值为:
" + add(i)+"<br>");
    out.print("变量"+i+"的原值为:"+i+",调用自减方法计算变量" + i + "的之后值为:
" + sub(i));
%>
</body>
</html>
```

在案例中分别调用了自增和自减两种方法，在调用前后比较变量的变化，效果显示如图 3-5 所示。

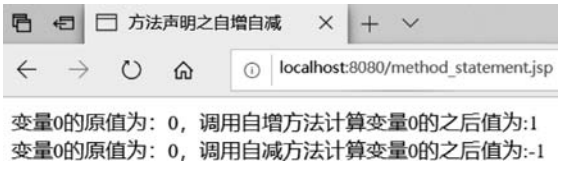

图 3-5　页面效果

3.3　Java 程序片段

3.3.1　程序片段定义

(1) 在“<%”和“%>”之间插入 Java 程序片段。一个 JSP 页面可以有许多程序片段，这些程序片段将被 JSP 引擎按顺序执行。

(2) 在 Web 容器处理 JSP 页面时执行，Java 程序片段通常会产生输出，并将输出发送

到客户的输出流里。

(3) 当多个客户请求一个 JSP 页面时,Java 程序片段将被执行多次,分别在不同的线程中执行,互不干扰。

3.3.2 程序片段的变量

(1) 因为 JSP 页面实际上是被编译成 Servlet 类执行的,所以声明中定义的变量是 Servlet 类的成员变量,各个用户共享成员变量,必须同步。

(2) 程序片段中定义的变量是局部变量,用户之间没有联系,每次调用页面,局部变量都被重新初始化。

程序片段定义的逻辑图如图 3-6 所示。

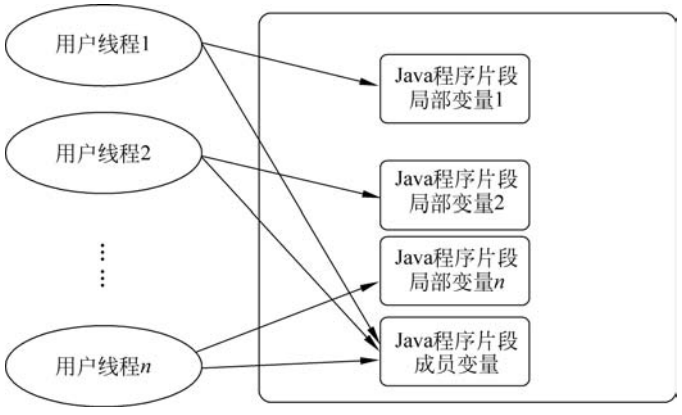

图 3-6 程序片段定义的逻辑图

Example3_3_share_member.jsp 中,使用一段简易计数器的代码来模拟 Java 程序片段中的各个用户共享成员变量,其中需要注意同步。

Example3_3_share_member.jsp

```
<%@ page import = "java.util.Date" %>
<%@ page language = "java" contentType = "text/html; charset = UTF - 8"
pageEncoding = "UTF - 8" %>
<!DOCTYPE html>
<html>
<head>
<meta charset = "UTF - 8">
<title>用户共享成员变量</title>
</head>
<body>
带你了解多个用户共享成员变量的概念!
<%!
  int i_shareMember = 0;

  synchronized void add() {
```

```
        i_shareMember++;
    }
%>
<%
    Date date = new Date();
    add();
    out.print(date + "当前时间段用户访问<br>");
    out.print("刷新之后,共享变量的值为" + this.i_shareMember + ".多个用户访问时其结果不
同<br>");
%>
</body>
</html>
```

实现的效果为多个用户在访问该变量时,保持各个用户之间的共享,即在第一个用户对成员变量进行有关操作后,影响后面用户对其调用的结果,通过时间的变化和共享变量的变化可以看出其区别。其页面显示效果如图 3-7 所示。

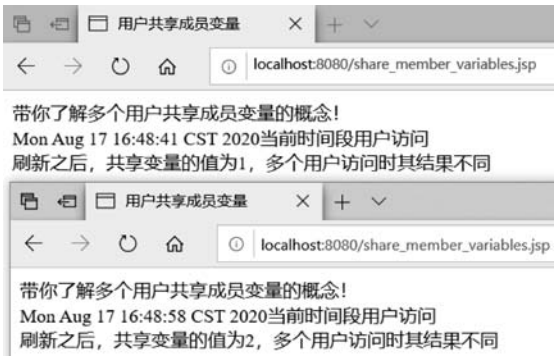

图 3-7　多用户共享成员变量案例页面显示

3.3.3　程序片段执行

一个 JSP 页面中的 Java 程序片段会按其在页面中的顺序被执行,而且某个 Java 程序片段中声明的局部变量在其后继的所有 Java 程序片段以及表达式内都有效。

可以将一个 Java 程序片段分割成几个 Java 程序片段,然后在这些 Java 程序片段之间再插入其他标记元素。程序片段的执行过程如图 3-8 所示。

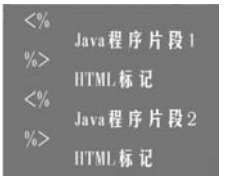

图 3-8　程序片段执行过程

通过 Example3_4_forward_code.jsp 了解分割的 Java 程序片段如何使用。案例使用 switch 选择语句对该模块进行介绍。

Example3_4_forward_code.jsp

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

```
<html>
<head>
<title>Java 程序片段分割</title>
</head>
<body>
<% //Math.random()是(0,1)的随机数
    int grade = (int) (Math.random() * 100);
    switch (grade/10){
        case 10:
        case 9:
%>
<jsp:forward page = "Example16_2_forward_excellent.jsp">
<jsp:param name = "number" value = "<% = grade %>"/>
</jsp:forward>
<%              break;
    case 8:
    case 7:
    case 6:
%>
<jsp:forward page = "Example16_3_forward_qualified.jsp">
<jsp:param name = "number" value = "<% = grade %>"/>
</ jsp:forward>
<%              break;
    default:
%>
<jsp:forward page = " Example16_4_forward_fail.jsp">
<jsp:param name = "number" value = "<% = grade %>"/>
</ jsp:forward>
<% }
%>
</body>
</html>
```

实现效果为产生一个 0～100 的随机数，通过 switch 选择语句判断该随机数的范围，效果如图 3-9 所示。

图 3-9 分割的 Java 程序片段案例页面展示

3.4 Java 表达式

在“<%=”和“%>”之间插入一个表达式，这个表达式必须能求值。表达式的值由服务器负责计算，并将计算结果用字符串形式发送到用户端显示。

例如，获取时间的表达式

```
<% = getDate() %>
```

其中需要注意“<%=”是一个完整的符号，“<%”和“=”之间不要有空格。

Example3_5_java_expression.jsp 运用表达式实现关于时间的类的运用手法。

Example3_5_java_expression.jsp

```
<% @ page import = "java.util.Calendar" %>
<% @ page language = "java" contentType = "text/html; charset = UTF - 8"
         pageEncoding = "UTF - 8" %>
<! DOCTYPE html >
< html >
< head >
< meta charset = "UTF - 8">
< title > Java 表达式</title >
</head >
< body >
<%
    Calendar calendar = Calendar.getInstance();
%>
现在是<% = calendar.get(Calendar.YEAR) %>年
<% = calendar.get(Calendar.MONTH) + 1 %>月
< br >当前时间为<% = calendar.get(Calendar.HOUR_OF_DAY) %>:<% = calendar.get(Calendar.
MINUTE) %>
</body >
</html >
```

页面的实现效果如图 3-10 所示。

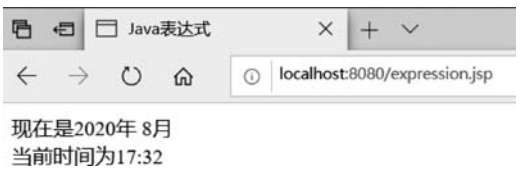

图 3-10　Java 表达式案例页面显示

3.5　JSP 注释

（1）HTML 注释格式：

```
<!-- 注释内容 -->
```

（2）JSP 注释格式：

```
<% -- 注释内容 -- %>
```

JSP 注释写在 JSP 程序中，但不发送给客户。

(3) Scriptlets 中的注释。由于 Scriptlets 包含的是 Java 代码,所以 Java 中的注释规则在 Scriptlets 中也适用,常用的 Java 注释使用//表示单行注释,使用/* */表示多行注释。

3.6 JSP 指令标记

在线视频

3.6.1 标记的种类

JSP 中主要有两种指令标记:page 指令标记和 include 指令标记。

3.6.2 page 指令标记

1. page 标记

(1) page 指令与其书写的位置无关,习惯上把 page 指令写在 JSP 页面的最前面。

(2) 可以用一个 page 指令指定多个属性的值,也可以使用多个 page 指令分别为每个属性指定值。

```
<%@ 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" %>
```

2. page 指令标记:contentType 属性

(1) contentType 属性值确定 JSP 页面响应的 MIME 类型和 JSP 页面字符的编码。属性值的一般形式是“MIME 类型”或“MIME 类型;charset=编码”。

例如:

```
<%@ page contentType = "application/msword" %>
```

(2) 如果不使用 page 指令为 contentType 指定一个值,那么 contentType 默认值是“text/html;charset=ISO-8859-1”。

注意:

不允许两次使用 page 指令给 contentType 属性指定不同的属性值。

3. page 指令标记:language 属性

language 属性定义 JSP 页面使用的脚本语言,该属性的值目前只能取“java”。

例如:

```
<%@ page language = "java" %>
```

4. page 指令标记:import 属性

目的是为 JSP 页面引入 Java 运行环境提供的包中的类,这样就可以在 JSP 页面的程序片段部分、变量及函数声明部分、表达式部分使用包中的类。

```
<%@ page import = "java.io.*", "java.util.Date" %>
```

注意：

JSP 页面默认 import 属性已经有 java.lang.*、javax.Servlet.*、javax.Servlet.jsp.*、javax.Servlet.http.* 等值。

5. page 指令标记：session 属性

session 属性用于设置是否需要使用内置的 session 对象。

session 的属性值可以是 true 或 false。session 属性默认的属性值是 true。

6. page 指令标记：buffer 属性

内置输出流对象 out 负责将服务器的某些信息或运行结果发送到用户端显示。buffer 属性用来指定 out 设置的缓冲区的大小或不使用缓冲区。

例如：

```
<%@ page buffer = "24kb" %>
```

注意：

buffer 属性的默认值是 8kb。buffer 属性可以取值“none”，即设置 out 不使用缓冲区。

7. page 指令标记：autoFlush 属性

(1) autoFlush 属性指定 out 的缓冲区被填满时，缓冲区是否自动刷新。

(2) autoFlush 属性的默认值是 true。

(3) 当 autoFlush 属性取值 false 时，如果 out 的缓冲区填满，就会出现缓存溢出异常。当 buffer 的值是“none”时，autoFlush 的值就不能设置成 false。

8. page 指令标记：isThreadSafe 属性

(1) isThreadSafe 属性用来设置 JSP 页面是否可多线程访问。

(2) 当 isThreadSafe 属性值设置为 true 时，JSP 页面能同时响应多个用户的请求；当 isThreadSafe 属性值设置成 false 时，JSP 页面同一时刻只能响应一个用户的请求，其他用户须排队等待。

注意：

isThreadSafe 属性的默认值是 true。

9. page 指令标记：info 属性

info 属性的属性值是一个字符串，其目的是为 JSP 页面准备一个常用且可能要经常修改的字符串。

例如：

```
<%@ page info = "we are students" %>
```

注意：

(1) 可以在 JSP 页面中使用方法 getServletInfo()；获取 info 属性的属性值。

(2) 当 JSP 页面被转译成 Java 文件时，转译成的类是 Servlet 的一个子类，所以在 JSP 页面中可以使用 Servlet 类的方法 getServletInfo()。

图 3-11 为 page 标记各个属性的详细介绍。

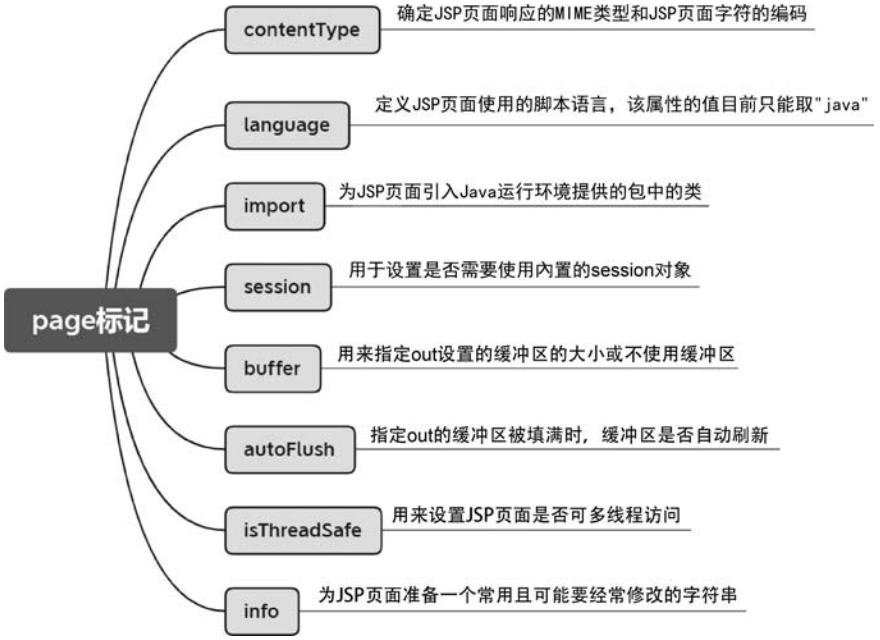

图 3-11　page 标记各属性介绍

3.6.3　include 指令标记

（1）include 指令标记的作用是在 JSP 页面出现该指令的位置处静态插入一个文件，实现代码的复用。

例如：

```
<%@ include file = "文件相对地址" %>
```

（2）一个 JSP 页面中的 include 指令的数量不受限制。

（3）静态插入，就是当前 JSP 页面和插入的文件合并成一个新的 JSP 页面，然后 JSP 引擎再将这个新的 JSP 页面转译成 Java 文件。

下面通过 Example3_6_include_code.jsp 了解其使用方法。

Example3_6_include_code.jsp

```
<%@ page language = "java" contentType = "text/html; charset = UTF - 8"
        pageEncoding = "UTF - 8" %>
<!DOCTYPE html>
<html>
<head>
<meta charset = "UTF - 8">
<title> include 动作案例</title>
</head>
<body>
使用 include 引入同级目录下的文件 Java 表达式，即引入 Example3_5_java_expression.jsp!<br>
<%! int i_shareMember = 0;   //共享变量 %>
<%
```

```
    out.print("刷新之后,成员变量的值为" + this.i_shareMember++ + ",其结果发生改变<br>");
 %>
<p>以下是引入 include 命令后的页面内容</p>
<% @ include file = " Example3_5_java_expression.jsp" %>
</body>
</html>
```

实现效果如图 3-12 所示,引入同级目录下的文件 Example3_5_java_expression.jsp。

使用 include 指令可以把一个复杂的 JSP 页面分成若干简单的部分,如要更改页面时,只需更改对应的部分就行了,页面布局如图 3-13 所示。

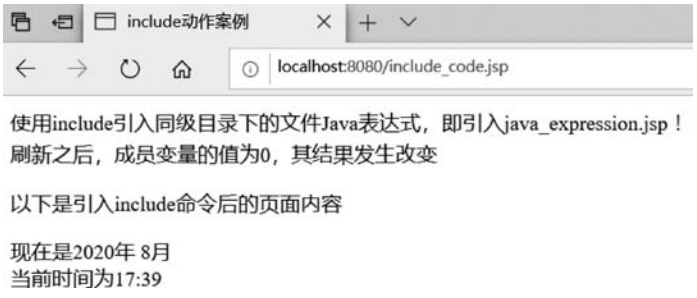

图 3-12　include 动作标记案例页面显示

图 3-13　页面布局

3.7　JSP 动作标记

3.7.1　标记的种类

JSP 动作标记类型如图 3-14 所示,具体的动作标记的作用和定义在后面的内容有具体介绍。

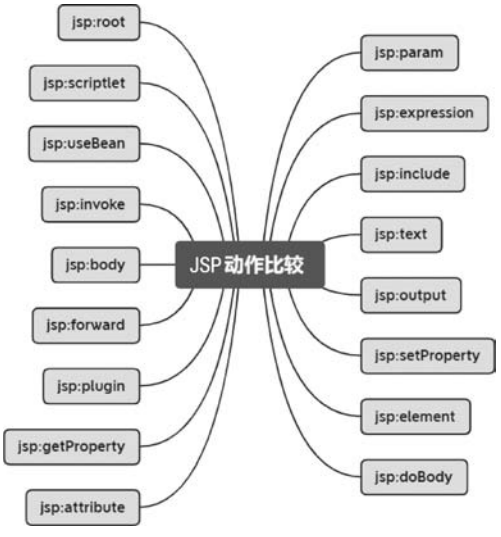

图 3-14　JSP 动作标记

3.7.2 include 动作标记

(1) include 动作标记告诉 JSP 页面动态包含一个文件,即 JSP 页面运行时才将文件加入。

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

或

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

(2) 如果是普通的文本文件,就将文件的内容发送到用户端,由浏览器负责显示;如果是 JSP 文件,JSP 引擎就执行这个文件,然后将执行的结果发送到用户端浏览器显示。

3.7.3 param 动作标记

(1) param 标记以“名字-值”对的形式为其他标记提供附加信息。

(2) param 动作标记语法格式为

```
<jsp:param name = "名字" value = "指定给 param 的值">
```

注意:

param 标记不能独立使用,需作为 jsp:include、jsp:forward、jsp:plugin 标记的子标记来使用。

(3) 当该标记与 jsp:include 动作标记一起使用时,可以将 param 标记中的值传递到 include 动作标记要加载的文件中去,被加载的 JSP 文件可以使用 Tomcat 服务器提供的 request 内置对象获取 include 动作标记的 param 子标记中 name 属性所提供的值。

param 标记的逻辑图如图 3-15 所示。

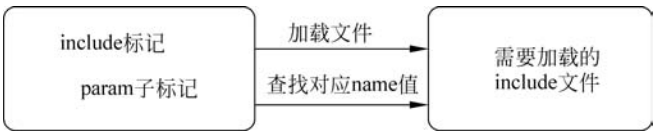

图 3-15 逻辑图

Example3_7_a_param.jsp 和 Example3_7_b_param_pythagorean.jsp 代码如下。

Example3_7_a_param.jsp

```
<%@ page language = "java" contentType = "text/html; charset = UTF - 8"
                pageEncoding = "UTF - 8" %>
<!DOCTYPE html>
<html>
<head>
<meta charset = "UTF - 8">
<title>判断勾股定理</title>
</head>
```

```
<body>
<% double a = 4,b = 4,c = 5;
 %>
<br>判断三边为<% = a %>,<% = b %>,<% = c %>的三角形是否满足勾股定理.
<jsp:include page = "Example3_7_b_param_pythagorean.jsp">
<jsp:param name = "sideA" value = "<% = a %>"/>
<jsp:param name = "sideB" value = "<% = b %>"/>
<jsp:param name = "sideC" value = "<% = c %>"/>
</jsp:include>
</body>
</html>
```

Example3_7_b_param_pythagorean.jsp

```
<%@ page language = "java" contentType = "text/html; charset = UTF - 8"
        pageEncoding = "UTF - 8" %>
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<%! public boolean getArea(double a, double b, double c) {
    if (a * a + b * b == c * c || a * a + c * c == b * b || b * b + c * c == a * a) {
        return true;
    } else {
        return false;
    }
}
 %>
<%
    String sideA = request.getParameter("sideA");
    String sideB = request.getParameter("sideB");
    String sideC = request.getParameter("sideC");
    double a = Double.parseDouble(sideA);
    double b = Double.parseDouble(sideB);
    double c = Double.parseDouble(sideC);
 %>
<br><b>我是被加载的文件,负责计算三角形是否满足勾股定理<br>
    三角形:<% if(getArea(a,b,c)){
 %>
<h4>满足勾股定理</h4>
<% } else {
 %><h4>不满足勾股定理</h4>
<% }
 %>
</body>
</html>
```

param 动作标记案例的页面显示效果如图 3-16 所示。

图 3-16 param 动作标记案例页面展示

3.7.4 forward 动作标记

```
<jsp:forward page="要转向的页面" />
```

或

```
<jsp:forward page="要转向的页面">
    param 子标记
</jsp:forward>
```

该指令的作用是：从该指令处停止当前页面的执行，而转向执行 page 属性指定的 JSP 页面。

也可以使用 param 动作标记作为子标记传送信息，要转向的 JSP 页面用 request 内置对象获取 param 子标记中 name 属性所提供的值。

注意：

(1) 当 forward 动作标记不需要 param 子标记时，必须使用第一种形式。

(2) 当前页面使用 forward 动作标记转向后，尽管用户看到了转向后的页面的效果，但浏览器地址栏中显示的仍然是转向前的 JSP 页面的 URL 地址。因此，如果刷新浏览器的显示，将再次执行当前浏览器地址栏中显示的 JSP 页面。

Example3_8_a_forward_code.jsp 等代码如下。

Example3_8_a_forward_code.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<html>
<head>
<title>Java 程序片段分割</title>
</head>
<body>
<% //Math.random()是(0,1)的随机数
    int grade = (int) (Math.random() * 100);
    switch (grade/10){
        case 10:
        case 9:
%>
<jsp:forward page="Example3_8_b_forward_excellent.jsp">
<jsp:param name="number" value="<%= grade %>"/>
```

```
</jsp:forward>
<%              break;
    case 8:
    case 7:
    case 6:
%>
<jsp:forward page="Example3_8_c_forward_qualified.jsp">
<jsp:param name="number" value="<%=grade%>"/>
</jsp:forward>
<%              break;
    default:
%>
<jsp:forward page="Example3_8_d_forward_fail.jsp">
<jsp:param name="number" value="<%=grade%>"/>
</jsp:forward>
<% }
%>
</body>
</html>
```

Example3_8_b_forward_excellent.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
<title>成绩优秀等次</title>
</head>
<body>
<%
    String appraise = request.getParameter("number");
%>
<h4>你的成绩为:<%=appraise%></h4>
<h4>你真优秀,希望你继续保持哦!</h4>
</body>
</html>
```

Example3_8_c_forward_qualified.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
<title>成绩合格等次</title>
</head>
<body>
<%
    String appraise = request.getParameter("number");
%>
<h4>你的成绩为:<%=appraise%></h4>
<h4>你真棒,希望你继续努力哦!加油～</h4>
```

```
</body>
</html>

Example3_8_d_forward_fail.jsp

<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
<title>成绩不合格等次</title>
</head>
<body>
<%
    String appraise = request.getParameter("number");
%>
<h4>你的成绩为:<% =appraise%></h4>
<h4>不要灰心,相信自己,继续加油!</h4>
</body>
</html>
```

forward 动作标记案例的页面显示如图 3-17 所示。

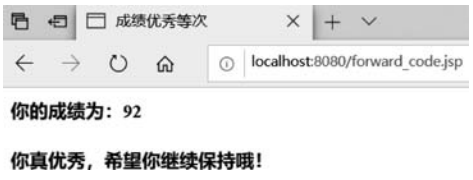

图 3-17 forward 动作标记案例页面展示

3.7.5 useBean 动作标记

该标签用于设置 JSP 使用的 JavaBean 的属性,此操作与 useBean 协作,用来设置 Bean 的简单属性和索引属性。

```
<jsp:useBean id="id" scope="page|request|session|application" typeSpec/>
```

其中,id 表示实例,scope 表示此对象可以使用的范围。

3.7.6 setProperty 动作标记

此操作与 useBean 协作,用来设置 Bean 的简单属性和索引属性。

<jsp:setProperty>标签使用 Bean 给定的 setXXX()方法,在 Bean 中设置一个或多个属性值。利用<jsp:setProperty>设置属性值有多种方法。

```
<jsp:setProperty name="Bean 的变量名" property="Bean 的属性名" value="Bean 的属性值" />
<!-- 或者将 setProperty 放在 useBean 的标签体中 -->
<jsp:useBean id="user" class="model.User">
<jsp:setProperty name="user" property="name" value="Bob"/>
</jsp:useBean>
```

3.7.7 getProperty 动作标记

jsp:getProperty 的操作是对 jsp:setProperty 操作的补充,它用来访问一个 Bean 的属性。

```
<jsp:getProperty name="userSession" property="name"/>
```

3.8 上机案例

本章的知识点已全部结束,读者在学习本章内容时是否遇到了难以解决的问题呢? 本节通过结合上机案例对本章知识点做针对性复习和补充,在课余时间也不要忘了回顾本章内容,在项目开发过程中本章内容也是至关重要的。

在本章上机案例中使用 JSP 动作标记完成一个网站主界面的部分展示,只需了解其大致结构即可。当我们访问一个网站的时候,大致可以将页面分为三个部分,分别为上中下部分,上部分为网站会员登录、注册和网站 logo 等信息,中部分为网站的主要信息,其中包含网站的介绍、产品展示等信息,下部分为网站的版权等信息。了解了其大致的结构之后可以使用 JSP 动作标记来简单模仿该功能。参考代码如下。

Example3_9_a_exam.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
        pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>上机案例</title>
</head>
<body>
<p>欢迎访问本网站!</p>
<h3>这是主界面内容,其中包含该网站的主要信息</h3>
<jsp:include page="Example3_9_b_exam.jsp"></jsp:include>
</body>
</html>
```

Example3_9_b_exam.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
        pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>上机案例</title>
</head>
<body>
<p>----------------------------------------</p>
<p>此处为网站底部的版权等信息</p>
```

```
<b>版权?2018 - 2020 </b>
</body>
</html>
```

小结

(1) JSP 页面：包括普通的 HTML 标记(客户端浏览器执行)、JSP 标记、成员变量和方法声明、Java 程序片段、Java 表达式(JSP 引擎处理并将结果发送给用户浏览器)。

(2) 成员变量为所有用户共享，任何用户对成员变量的操作都会影响其他用户，synchronized 关键字保证一次只有一个线程执行。

(3) 多用户访问 JSP 页面，其程序片段会被执行多次，分别在不同线程中，其局部变量互不干扰。

(4) page 指令标记用来定义整个 JSP 页面的一些属性，常用的有 contentType 和 import。

(5) include 指令标记在编译阶段就处理所需要的文件，被处理的文件在逻辑与语法上依赖于当前 JSP 页面，优点是速度快；include 动作标记是在 JSP 页面运行时才处理文件，在逻辑与语法上独立于当前 JSP 页面，更加灵活。

本章小结可参考图 3-18。

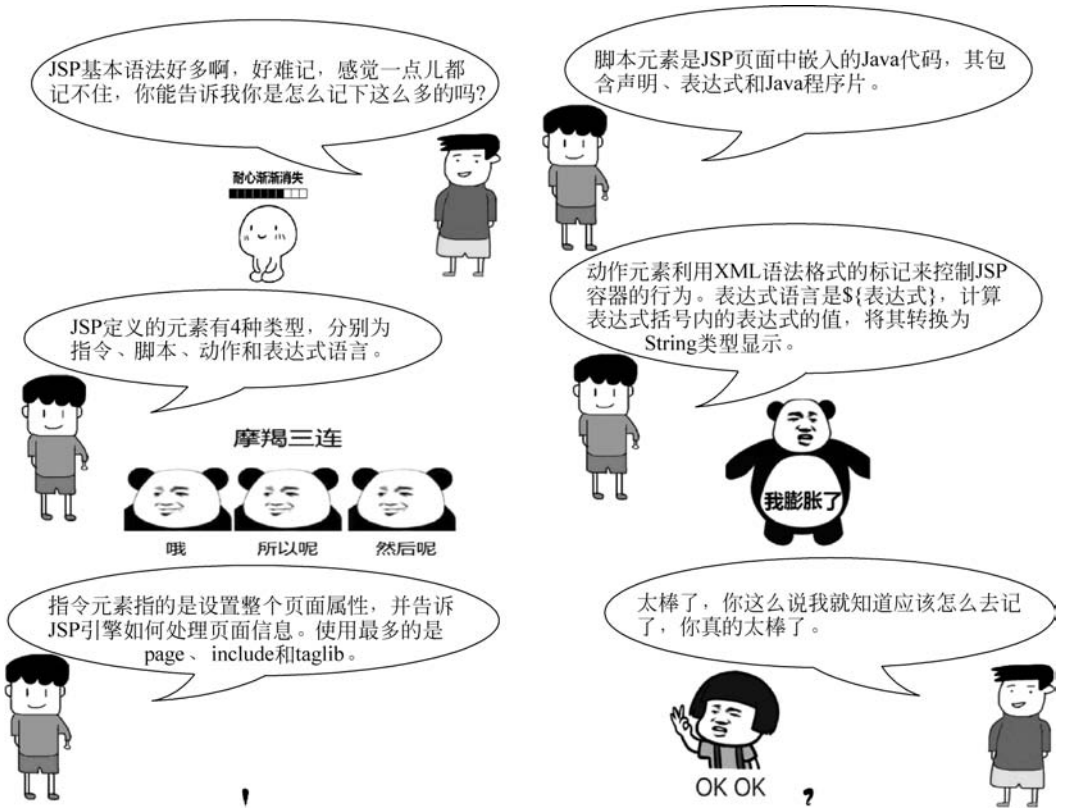

图 3-18 第 3 章小结

习题

1. 对于 JSP 中的 HTML 注释叙述正确的是(　　)。
   A. 发布网页时看不到,在源文件中也看不到
   B. 发布网页时看不到,在源代码中看得到
   C. 发布网页时能看到,在源文件中看不到
   D. 发布网页时能看到,在源文件中也能看到
2. 用户获取 Bean 属性的动作是(　　)。
   A. <jsp:userBean>　　B. <jsp:getProperty>
   C. <jsp:setProperty>　　D. <jsp:param>
3. JSP 程序中的注释有:_____、_____和_____。
4. JSP 的编译指令标记通常是指(　　)。
   A. Page 指令、Include 指令和 Taglib 指令
   B. Page 指令、Include 指令和 Plugin 指令
   C. Param 指令、Include 指令和 Plugin 指令
   D. Page 指令、Forward 指令和 Taglib 指令
5. 指令标记、JSP 动作标记统称为_____。
6. 在"<%!"和"%>"之间声明的变量称为_____,在其之间声明的方法称为_____。
7. JSP 页面的基本结构由哪几部分组成?
8. 简述 include 指令和<jsp:include>动作的异同。
9. 简述 page 指令、include 指令的作用。
10. 模仿案例 3_4 设计 JSP 代码实现 Java 程序片段。