第 3 章 JSP 技术基础

本章学习 JSP 各种元素的使用，其中包括脚本元素、隐含变量、指令，还将学习作用域变量、JavaBeans、MVC 设计模式和错误处理方法。

3.1 知识点总结

（1）JSP 脚本有三种：JSP 声明（<%! Java 声明%>）、小脚本（<% Java 代码 %>）和 JSP 表达式（<%= 表达式 %>）。

（2）在 JSP 的脚本中可以使用 9 个隐含变量，它们分别是 application、session、request、response、page、pageContext、out、config 和 exception 等。

（3）在 JSP 中可以使用的指令有三种类型：page 指令、include 指令和 taglib 指令。三种指令的语法格式如下：

```
<%@ page attribute-list %>
<%@ include attribute-list %>
<%@ taglib attribute-list %>
```

page 指令通知容器关于 JSP 页面的总体特性，include 指令实现把另一个文件（HTML、JSP 等）的内容包含到当前页面中，taglib 指令用来指定在 JSP 页面中使用标准标签或自定义标签的前缀与标签库的 URI。

（4）在 JSP 中可使用三种类型的动作。标准动作、JSTL 动作和自定义动作。下面是常用的标准动作：

- <jsp:include>动作用于包含另一个页面输出。
- <jsp:forward>动作将请求转发到指定页面。
- <jsp:useBean>动作用来在 JSP 页面中查找或创建一个 bean 实例。
- <jsp:setProperty>动作用来给 bean 实例的属性赋值。
- <jsp:getProperty>动作用来检索并向输出流中打印 bean 的属性值。

（5）表达式语言 EL，它是一种数据表示语言，例如，${applicationScope.email}输出应用作用域中的 email 属性值。

（6）JSP 页面本质上也是 Servlet，但若仅实现表示逻辑编写 JSP 页面要比编写 Servlet 容易。JSP 页面也在容器中运行，当 JSP 页面第一次被访问时，Web 容器解析 JSP 文件并将其转换成页面实现类。接下来，Web 容器编译该类并将其装入内存，然后与其他 Servlet 一样执行并将其输出结果发送到客户端。

（7）在 JSP 页面中有 4 个作用域对象，它们的类型分别是 ServletContext、HttpSession、

HttpServletRequest 和 PageContext，这 4 个作用域分别称为应用（application）作用域、会话（session）作用域、请求（request）作用域和页面（page）作用域。

（8）在 Java Web 开发中常用 JavaBeans 来存放数据、封装业务逻辑等，在 JSP 页面中使用 JavaBeans 主要是通过三个 JSP 标准动作实现的。

（9）MVC 设计模式称为模型-视图-控制器模式。模型用 JavaBeans 实现，视图用 JSP 实现，控制器用 Servlet 或过滤器实现。

（10）在 Java Web 开发中有多种错误处理方法：声明式错误处理和编程式错误处理。

3.2 实 训 任 务

【实训目标】

学会 JSP 页面各种元素的使用，理解页面实现类，使用包含指令和包含动作，掌握 JavaBeans 和作用域概念。

任务 1　学习 JSP 页面如何转换成页面实现类

本任务学习 JSP 页面元素如何转换成页面实现类。

（1）在 Eclipse 中新建一个 jsp-demo 动态 Web 项目，在项目的 WebContent 目录中新建 today-date.jsp 页面，代码如下：

```
<%@page import="java.time.LocalDate" %>
<%@page contentType="text/html;charset=UTF-8" %>
<html>
<head><title>显示日期</title></head>
<%
   LocalDate today  = LocalDate.now();
%>
<body>
今天的日期是：<%=today%>
</body>
</html>
```

（2）启动浏览器，访问 today-date.jsp 页面，显示结果如图 3-1 所示。

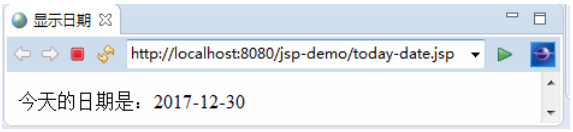

图 3-1　today-date.jsp 页面运行结果

（3）假设将 jsp-demo 项目部署到 Tomcat 服务器中。打开 Tomcat 安装目录的 \work\Catalina\localhost\jsp-demo\org\apache\jsp 目录中的 today-date_jsp.java 文件，查看隐含对象是如何定义的，完成下面的填空。

JSP 页面转换后定义的类名为（　　）。
该类继承了哪个类？（　　）。
隐含对象 request 的类型为（　　）。
隐含对象 response 的类型为（　　）。
隐含对象 pageContext 的类型为（　　）。
隐含对象 session 的类型为（　　）。
隐含对象 application 的类型为（　　）。
隐含对象 config 的类型为（　　）。
隐含对象 out 的类型为（　　）。
隐含对象 page 的类型为（　　）。
是作用域对象的包括（　　）。
（4）在页面实现类中找到 today 变量的声明位置。

任务 2　学习使用包含指令和包含动作实现页面布局

本任务包括 4 个文件 header.jsp、body.jsp、footer.jsp 和 main.jsp。在 main.jsp 文件中使用 include 指令包含其他页面实现页面布局。

（1）在 jsp-demo 项目的 WebContent 目录中创建 header.jsp 文件，它实现页面页眉部分。

```
<%@ page contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<p style="color:#ff0000;">
新世纪 网上书店</p>
<hr />
```

（2）在 WebContent 目录中创建 body.jsp 文件，它实现页面主体部分。

```
<%@ page contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<table border=0 cellspacing=5 cellpadding=5 width="100%">
  <tr><td>
    <p align="center"><b>欢迎光临新世纪网上书店！</b></p>
   </td>
  </tr>
  <tr>
   <td>
    <p align="center"><b><a href="/bookstore/catalog">开始购买图书</a></b>
   </td>
  </tr>
</table>
```

（3）在 WebContent 目录中创建 footer.jsp 文件，它实现页面的页脚部分。

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

```
    pageEncoding="UTF-8"%>
<hr />
<center>版权所有 &copy; 2018 新世纪网上书店, Inc.</center>
```

（4）在 WebContent 目录中创建 main.jsp 文件，它是主页面。其中使用 include 指令包含其他部分的文件。

```
<%@ page contentType="text/html; charset=UTF-8"
         pageEncoding="UTF-8"%>
<html>
<head><title>新世纪在线书店</title></head>
<body>
<%@ include file="header.jsp" %>
<%@ include file="body.jsp" %>
<%@ include file="footer.jsp" %>
</body>
</html>
```

（5）在浏览器中访问 main.jsp 文件，结果如图 3-2 所示。

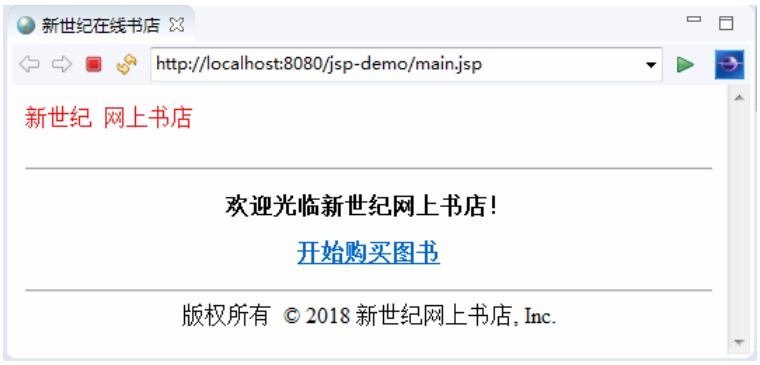

图 3-2　main.jsp 页面运行结果

（6）修改上面程序，使用<jsp:include>动作实现页面布局。

任务 3　学习在 Servlet 和 JSP 页面中使用 JavaBeans 对象

本任务在 JSP 页面 inputProduct.jsp 中输入商品信息，将请求转到 ProductServlet，创建一个 Product 对象，然后将请求转发到 dispalyProduct.jsp 页面，在其中使用<jsp:getProperty>动作输出信息。

（1）在 jsp-demo 项目的 com.demo 包中创建 Product 类，它是 JavaBeans 类，用于存放商品信息。

```
package com.demo;
public class Product {
    private String id;          //商品号
    private String name;        //商品名
```

```
    private double price;           //价格
    public Product() {
        super();
    }
    public Product(String id, String name, double price) {
        super();
        this.id = id;
        this.name = name;
        this.price = price;
    }
    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public double getPrice() {
        return price;
    }
    public void setPrice(double price) {
        this.price = price;
    }
}
```

（2）在 WebContent 目录中创建 input-product.jsp 页面，用于输入商品信息，代码如下：

```
<%@ page contentType="text/html; charset=UTF-8" %>
<html>
<head> <title>输入商品信息</title></head>
<body>
<h4>输入商品信息</h4>
<form action = "product-servlet" method = "post">
    <label>商品号：<input type="text" name="id" ></label><br>
    <label>商品名：<input type="text" name="name"></label><br>
    <label>价格：<input type="text" name="price" ></label><br>
    <label><input type="submit" value="确定" >
        <input type="reset" value="重置" ></label>
</form>
</body>
</html>
```

（3）在 src 的 com.demo 包中创建 ProductServlet，在其中检索用户提交的商品信息，构建 Product 对象并存储到请求作用域中，将请求转发到 display-product.jsp 页面。

```
package com.demo;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.WebServlet;
@WebServlet("/product-servlet")
public class ProductServlet extends HttpServlet {
    public void doPost(HttpServletRequest request,
                       HttpServletResponse response)
              throws java.io.IOException,ServletException {
        String id = request.getParameter("id");
        String name = request.getParameter("name");
        //修改请求参数的字符编码
        name = new String(name.getBytes("iso-8859-1"),"UTF-8");
        double price = Double.parseDouble(request.getParameter("price"));
        Product product = new Product(id,name,price);
        request.setAttribute("product", product);
        RequestDispatcher rd =
           request.getRequestDispatcher("/display-product.jsp");
        rd.forward(request,response);
    }
}
```

（4）在 WebContent 目录中创建 display-product.jsp 页面，用 JavaBeans 标准动作显示请求作用域的商品信息，代码如下：

```
<%@ page contentType="text/html; charset=UTF-8"
          pageEncoding="UTF-8"%>
<jsp:useBean id="product" class="com.demo.Product" scope="request" />
<html><head><title>商品信息</title></head>
<body>
    <table border="1">
    <caption>商品信息如下</caption>
    <tr>
    <td>商品号:</td>
    <td><jsp:getProperty name="product" property="id"/></td>
    </tr>
    <tr>
    <td>商品名:</td>
    <td><jsp:getProperty name="product" property="name"/></td>
    </tr>
    <tr>
    <td>价格:</td>
    <td><jsp:getProperty name="product" property="price"/></td>
```

```
    </tr>
    </table>
</body></html>
```

（5）访问 input-product.jsp 页面，在其中输入商品号 202，商品名“笔记本电脑”，价格输入 4200，最后显示的 display-product.jsp 页面如图 3-3 所示。

图 3-3　displayProduct.jsp 页面运行结果

3.3　思考与练习答案

1．下面左边一栏是 JSP 元素类型，右边是对应名称，请连线。

```
<% Float one = new Float(88.88) %>          指令
<%! int y = 3; %>                           EL表达式
<%@ page import="java.util.*" %>            声明
<jsp:include page="foo.jsp" />              小脚本
<%=pageContext.getAttribute("foo") %>       动作
email:${applicationScope.mail}              表达式
```

【答】

```
  <% Float one = new Float(88.88) %>        小脚本
  <%! int y = 3; %>                         声明
  <%@ page import="java.util.*" %>          指令
  <jsp:include page="foo.jsp" />            动作
  <%=pageContext.getAttribute("foo") %>     表达式
  email:${applicationScope.mail}            EL表达式
```

2．执行下面 JSP 代码输出结果是什么？（　　）

```
<% int x = 3; %>
<%! int x = 5; %>
<%! int y = 6; %>
x与y的和是：<%=x+y%>
```

A．x 与 y 的和是：8　　B．x 与 y 的和是：9

C．x 与 y 的和是：11　　D．发生错误

【答】 B。变量 x 被声明两次：一次是作为类的成员变量，因为使用了<%! int x = 5; %>语句，另一次是在_jspService()中声明的局部变量，因为使用的代码是<% int x = 3; %>。

3．下面 JSP 代码有什么错误？

```
<!% int i = 5; %>
<!% int getI() { return i; } %>
```

【答】 正确声明应为：

```
<%! int i = 5; %>
<%! int getI() { return i; } %>
```

4．假设 myObj 是一个对象的引用，m1()是该对象上一个合法的方法。下面的 JSP 结构哪个是合法的？（　　）

A．<% myObj.m1() %>　　B．<%=myObj.m1() %>

C．<% =myObj.m1() %>　　D．<% =myObj.m1(); %>

【答】 B 是合法的。注意，JSP 表达式中百分号和等号之间不能有空格。

5．说明下面代码是否是合法的 JSP 结构？（　　）

A．<%=myObj.m1(); %>

B．<% int x=4, y=5; %>
<%=x=y%>

C．<% myObj.m1(); %>

【答】 B、C。A 非法，等号表明它是 JSP 表达式，但表达式不能以分号结束。B 合法，<%=x=y%>将被转换成：

```
out.print(x=y);  //y的值5赋给x并将其打印输出
```

C 是合法的小脚本，因为在方法调用语句的后面有分号。即使方法返回一个值，它也是合法的，返回值将被忽略。

6．下面哪个 page 指令是合法的？（　　）

A．<% page language="java" %>　　B．<%! page language="java" %>

C．<%@ page language="java" %>　　D．<%@ Page language="java" %>

【答】 C。

7．下面的 page 指令哪个是合法的？（　　）

A．<%@ page import="java.util.* java.text.* " %>

B．<%@ page import="java.util.*", "java.text.* " %>

C．<%@ page buffer="8kb", session="false" %>

D．<%@ page import="com.manning.servlets.* " %>
<%@ page session="true" %>
<%@ page import="java.text.*" %>

E．<%@ page bgcolor="navy" %>

F．<%@ page buffer="true" %>

G．<%@ Page language='java' %>

【答】 D。选项 A 中 import 的属性值中应该有逗号。选项 B 的 import 属性值应该在一个字符串中指定。选项 C，属性之间不允许有逗号。选项 E，bgcolor 不是合法的属性名。选项 F，true 不是 buffer 属性合法值。选项 G，指令名、属性名和值都是大小写敏感的，Page 应为 page。

8．下面哪些是合法的 JSP 隐含变量？（　　）

A．stream　　B．context

C．exception　　D．listener

E．application

【答】 C，E。

9．下面是 JSP 生命周期的各个阶段，正确的顺序应该是（　　）。

① 调用_jspService()

② 把 JSP 页面转换为 Servlet 源代码

③ 编译 Servlet 源代码

④ 调用 jspInit()

⑤ 调用 jspDestroy()

⑥ 实例化 Servlet 对象

【答】 ②③⑥④①⑤。

10．有下面 JSP 页面，给出该页面每一行在转换的 Servlet 中的代码。

```
<html><body>
  <% int count = 0 ;%>
  The page count is now:
  <%= ++count %>
</body></html>
```

【答】 转换结果如下：

```
out.write("<html><body>\r\n");
int count = 0 ;
out.write("   The page count is now:\r\n");
out.print( ++count );
out.write("</body></html>\r\n");
```

11．有下列名为 counter.jsp 的页面，其中有三处错误。执行该页面，从浏览器输出中找出错误，修改错误直到页面执行正确。

```
<%@ Page contentType="text/html;charset=UTF-8" %>
<html><body>
  <%! int count = 0 %>
  <% count++; %>
  该页面已被访问 <%= count ;%> 次。
</body></html>
```

【答】

```
Page改为page                    //page的p应为小写
<%! int count = 0 %>            //声明缺少分号
<% count++; %>                  //去掉分号
```

12．以下关于 JSP 生命周期的方法，哪个是正确的？（　　）

A．只有 jspInit()可以被覆盖

B．只有 jspdestroy()可以被覆盖

C．jspInit()和 jspdestroy()都可以被覆盖

D．jspInit()、_jspService()和 jspdestroy()都可以被覆盖

【答】 C。_jspService()方法不能被覆盖。

13．下面哪个 JSP 标签可以在请求时把另一个 JSP 页面的结果包含到当前页面中？（　　）

A．<%@ page import %>　　B．<jsp:include>

C．<jsp: plugin>　　D．<%@ include %>

【答】 B。<jsp:include>动作实现动态包含，<%@ include %>实现静态包含。

14．在一个 JSP 页面中把请求转发到 view.jsp 页面，下面哪个是正确的？（　　）

A．<jsp:forward file="view.jsp" />　　B．<jsp:forward page="view.jsp" />

C．<jsp:dispatch file="view.jsp" />　　D．<jsp:dispatch page="view.jsp" />

【答】 B。<jsp:forward>动作用于实现请求转发，它只有一个 page 属性。

15．当 Servlet 处理请求发生异常时，使用下面哪个方法可向浏览器发送错误消息？（　　）

A．HttpServlet 的 sendError(int errorCode)方法

B．HttpServletRequest 的 sendError(int errorCode)方法

C．HttpServletResponse 的 sendError(int errorCode)方法

D．HttpServletResponset 的 sendError(String errorMsg)方法

【答】 C。

16．在部署描述文件中<exception-type>元素包含在哪个元素中？（　　）

A．<error>　　B．<error-mapping>

C．<error-page>　　D．<exception-page>

【答】 C。<error-page>元素包含<error-code>、<exeption-type>和<location>元素。

17．MVC 设计模式不包括下面的（　　）。

A．模型　　B．视图

C．控制器　　D．数据库

【答】 D。其中，M 表示模型、V 表示视图、C 表示控制器。

18．什么是 MVC 设计模式？简述实现 MVC 设计模式的一般步骤。

【答】 MVC 模式称为模型-视图-控制器模式。该模式将 Web 应用的组件分为模型、视图和控制器，每种组件完成各自的任务。该模型将业务逻辑和数据访问从表示层分离出来。实现 MVC 模式的一般步骤：① 定义 JavaBeans 表示数据；② 使用 Servlet 处理请求；

③ 将结果存储在作用域对象中；④ 将请求转发到 JSP 页面；⑤ 最后在 JSP 页面中从 JavaBeans 中取出数据。

19．简述声明式错误处理和编程式错误处理。

【答】 声明式错误处理可以使用 page 指令的 errorPage 属性指定一个错误处理页面，通过 page 指令的 isErrorPage 属性指定页面是错误处理页面。此外，还可以在 web.xml 文件中为整个 Web 应用配置错误处理页面。

编程式错误处理是指在 Servlet 中将代码包含在 try-catch 块中，在异常发生时通过 catch 块将错误消息发送给浏览器。