Web应用开发
之
JSP技术模型

赵小敏
浙江工业大学计算机科学与技术学院

本节内容

- 3.6 作用域对象
- 3.7 JSP组件包含

3.6 作用域对象

- 在JSP页面中有四个作用域对象，它们的类型分别是ServletContext、HttpSession、HttpServletRequest和PageContext，这四个作用域分别称为：
  - 应用（application）作用域
  - 会话（session）作用域
  - 请求（request）作用域
  - 页面（page）作用域

  这些作用域定义了对象存在性和从JSP页面和Servlet中的可访问性。

3.6 作用域对象

作用域名	对应的对象	存在性和可访问性
应用作用域	application	在整个Web应用程序有效
会话作用域	session	在一个用户会话范围内有效
请求作用域	request	在用户的请求和转发的请求内有效
页面作用域	pageContext	只在当前的页面（转换单元）内有效

- 属性存取方法:
  - void setAttribute(String name, Object value)方法
  - Object getAttribute(String name)方法

3.6.1 应用作用域

- 存储在应用作用域的对象可被Web应用程序的所有组件共享并在应用程序生命期内都可以访问。这些对象是通过ServletContext实例作为“属性/值”对维护的。
- 在JSP页面中，该实例可以通过隐含对象application访问。因此，要在应用程序级共享对象，可以使用ServletContext接口的setAttribute()和getAttribute()。

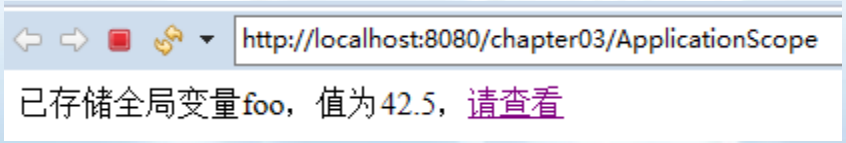
3.6.1 应用作用域

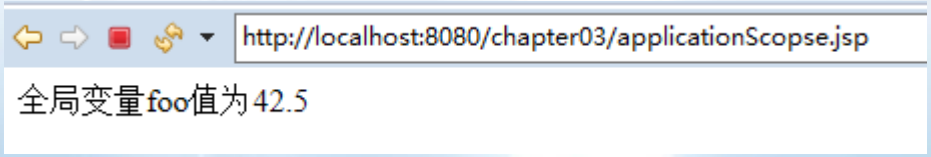
- 在Servlet使用下面代码将对象存储在应用作用域中：

```
Float one = new Float(42.5) ;
ServletContext context =getServletContext();
context.setAttribute("foo", one);
```

- 在JSP页面中就可使用下面代码访问context中数据：

```
<%=application.getAttribute("foo") %>
```





3.6.2 会话作用域

- 存储在会话作用域的对象可以被属于一个用户会话的所有请求共享并只能在会话有效时才可被访问。
- 这些对象是通过HttpSession类的一个实例作为“属性/值”对维护的。在JSP页面中该实例可以通过隐含对象session访问。
- 要在会话级共享对象，可以使用HttpSession接口的setAttribute()和getAttribute()。

3.6.2 会话作用域

- 在购物车应用中，用户的购物车对象就应该存放在会话作用域中，它在整个的用户会话中共享。

```
HttpSession session =request.getSession(true);
ShoppingCart cart = (ShoppingCart)session.getAttribute("cart");
if (cart == null) {
    cart = new ShoppingCart();
     // 将购物车存储到会话对象中
    session.setAttribute("cart", cart);
}
```

3.6.3 请求作用域

- 存储在请求作用域的对象可以被处理同一个请求的所有组件共享并仅在该请求被服务期间可被访问。这些对象是由HttpServletRequest对象作为“属性/值”对维护的。
- 在JSP页面中，该实例是通过隐含对象request的形式被使用的。
- 在Servlet中使用请求对象的setAttribute()将一个对象存储到请求作用域中，然后将请求转发到JSP页面，在JSP页面中通过脚本或EL取出作用域中的对象。

3.6.3 请求作用域

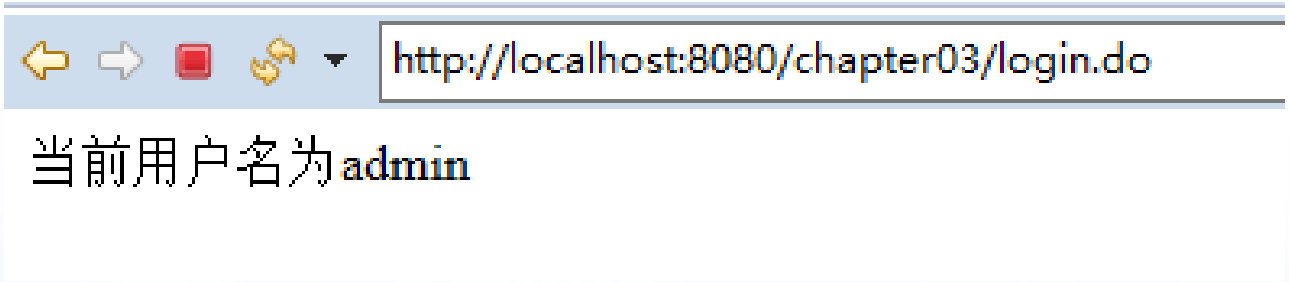
- 在Servlet中创建一个User对象并存储在请求作用域中，然后将请求转发到showUser.jsp页面。

```
User user = new User();
user.setName(request.getParameter("name"));
user.setPassword(request.getParameter("password"));
String uri="";
if(user.isValid()) {
request.setAttribute("user", user);
  uri="showUser.jsp";
}
else {
  uri="login.html";
}
RequestDispatcher rd =  request.getRequestDispatcher(uri);
rd.forward(request,response);
```

3.6.3 请求作用域

- showUser.jsp显示当前登录的用户信息。

```
<%
  User user = (User) request.getAttribute("user");
%>
当前用户名为<%=user.getName() %>
```



3.6.4 页面作用域

- 存储在页面作用域的对象只能在它们所定义的转换单元中被访问。这些对象是由PageContext抽象类的一个具体子类的一个实例通过属性/值对维护的。
- 在JSP页面中，该实例可以通过隐含对象pageContext访问。
- 为在页面作用域中共享对象，可以使用javax.servlet.jsp.PageContext定义的两个方法，其格式如下：

```
public void setAttribute(String name , Object value)
public Object getAttribute(String name )
```

3.6.4 页面作用域

- 设置一个页面作用域的属性：

```
<% Float one = new Float(42.5);%>
<% pageContext.setAttribute("foo", one );%>
```

- 获得一个页面作用域的属性：

```
<%=pageContext.getAttribute("foo")%>
```

3.6.4 页面作用域

- PageContext类中还定义了几个常量和其他属性处理方法，使用它们可以方便地处理不同作用域的属性。
- PageContext类定义的常量有：

✓ public static final int APPLICATION_SCOPE，表示application作用域。

✓ public static final int SESSION_SCOPE，表示session作用域。

✓ public static final int REQUEST_SCOPE，表示request作用域。

✓ public static final int PAGE_SCOPE，表示page作用域。

3.6.4 页面作用域

- PageContext类定义的方法有：

✓ public void setAttribute(String name, Object object, int scope)：在指定的作用域中设置属性。

✓ public Object getAttribute(String name, int scope)：返回在指定作用域中名为name的对象，没有找到则返回null。

✓ public Object findAttribute(String name)：查找指定名称的属性值。查找顺序为页面作用域、请求作用域、会话作用域（若有效）、应用作用域。

✓ public int getAttributesScope(String name)：返回给定属性的作用域。

3.6.4 页面作用域

- 使用pageContext获得一个应用作用域的属性：

```
Email is:
<%=pageContext.getAttribute("email",PageContext.APPLICATION_SCOPE)%>
```

- 上述代码等价于：

```
Email is:
<%=application.getAttribute("email")%>
```

- 使用pageContext，即使不知道作用域也可以查找一个属性

```
<%= pageContext.findAttribute("foo")%>
```

3.7 JSP组件包含

- 代码的可重用性是软件开发的一个重要原则
- 使用可重用的组件可提高应用程序的生产率和可维护性。
- JSP规范定义了一些允许重用Web组件的机制，其中包括在JSP页面中包含另一个Web组件的内容或输出。
- 两种实现方式：静态包含或动态包含。

3.7.1 静态包含：include指令

- 静态包含是在JSP页面转换阶段将另一个文件的内容包含到当前JSP页面中。使用JSP的include指令完成这一功能，它的语法为：

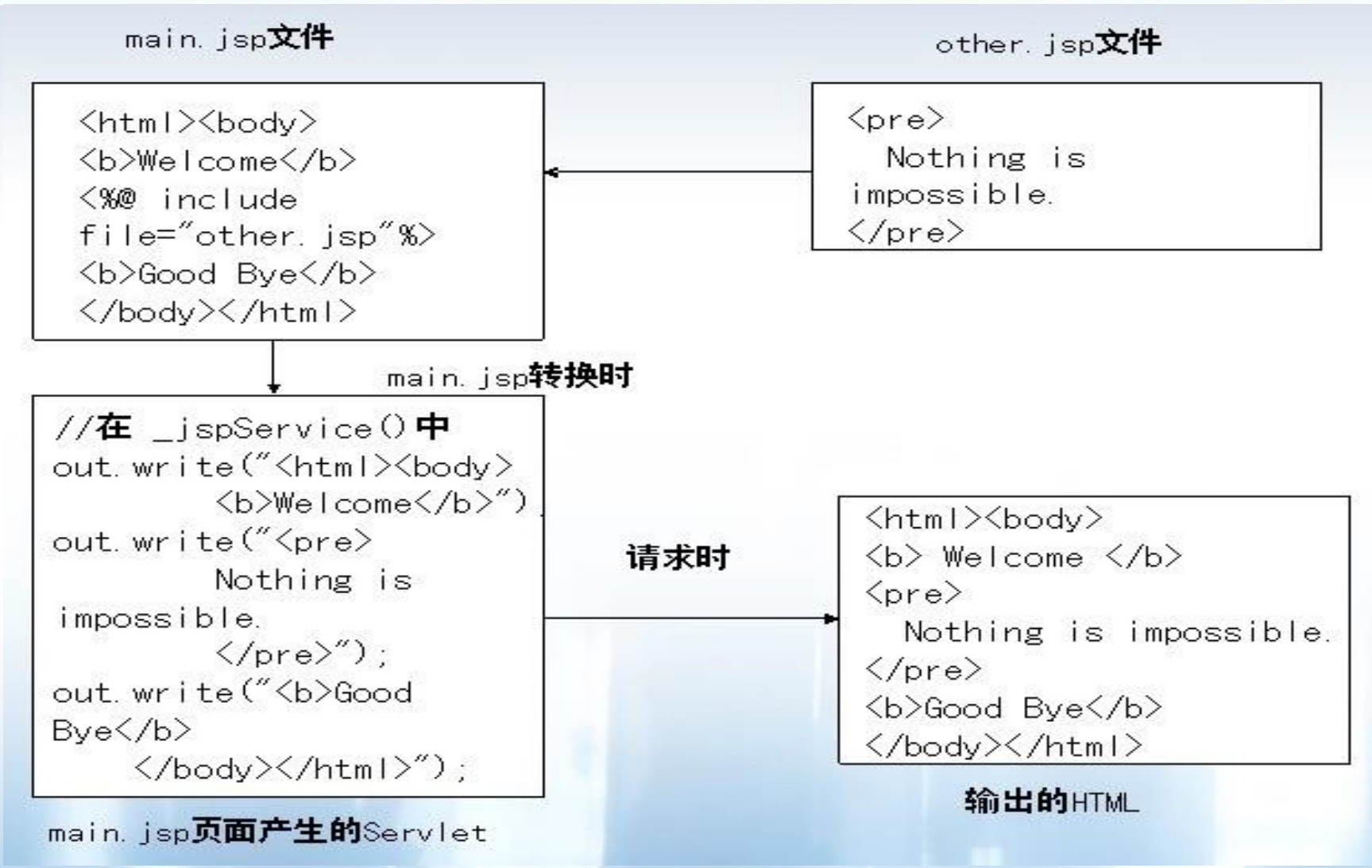
```
<%@ include file="relativeURL" %>
```

- file属性是include指令唯一的属性，它是指被包含的文件。

① 文件使用相对路径指定，相对路径或者以斜杠（/）开头，是相对于Web应用程序文档根目录的路径，或者不以斜杠开头，它是相对于当前JSP文件的路径。

② 被包含的文件可以是任何基于文本的文件，如HTML、JSP、XML文件，甚至是简单的TXT文件。

include指令的工作方式



1. 从被包含页面中访问变量

- 被包含JSP页面的代码成为主页面代码的一部分，因此，每个页面都可以访问在另一个页面中定义的变量,它们也共享所有的隐含变量

- 程序 hello.jsp和被包含页面response.jsp

2. 静态包含的限制

- 当使用include指令包含一个文件时，需要遵循下列几个规则：

  （1）在转换阶段不进行任何处理，这意味着file属性值不能是请求时表达式，因此下面的使用是非法的。

```
<%! String pageURL ="copyright.html"; %>
<%@ include file="<%= pageURL %>" %>
```

2. 静态包含的限制

（2）不能通过file属性值向被包含的页面传递任何参数，因为请求参数是请求的一个属性，它在转换阶段没有任何意义。下面例子中的file属性值是非法的。

```
<%@ include file="other.jsp?name=Hacker"%>
```

（3）被包含的页面可能不能单独编译。一般来说，最好避免这种依赖性，而使用隐含变量pageContext共享对象，通过使用pageContext的setAttribute()和getAttribute()实现。

3.7.2 动态包含：include动作

- 动态包含是通过JSP标准动作<jsp:include>实现的。该动作的格式如下：

```
<jsp:include page=“relativeURL”
        flush="true|false" />
```

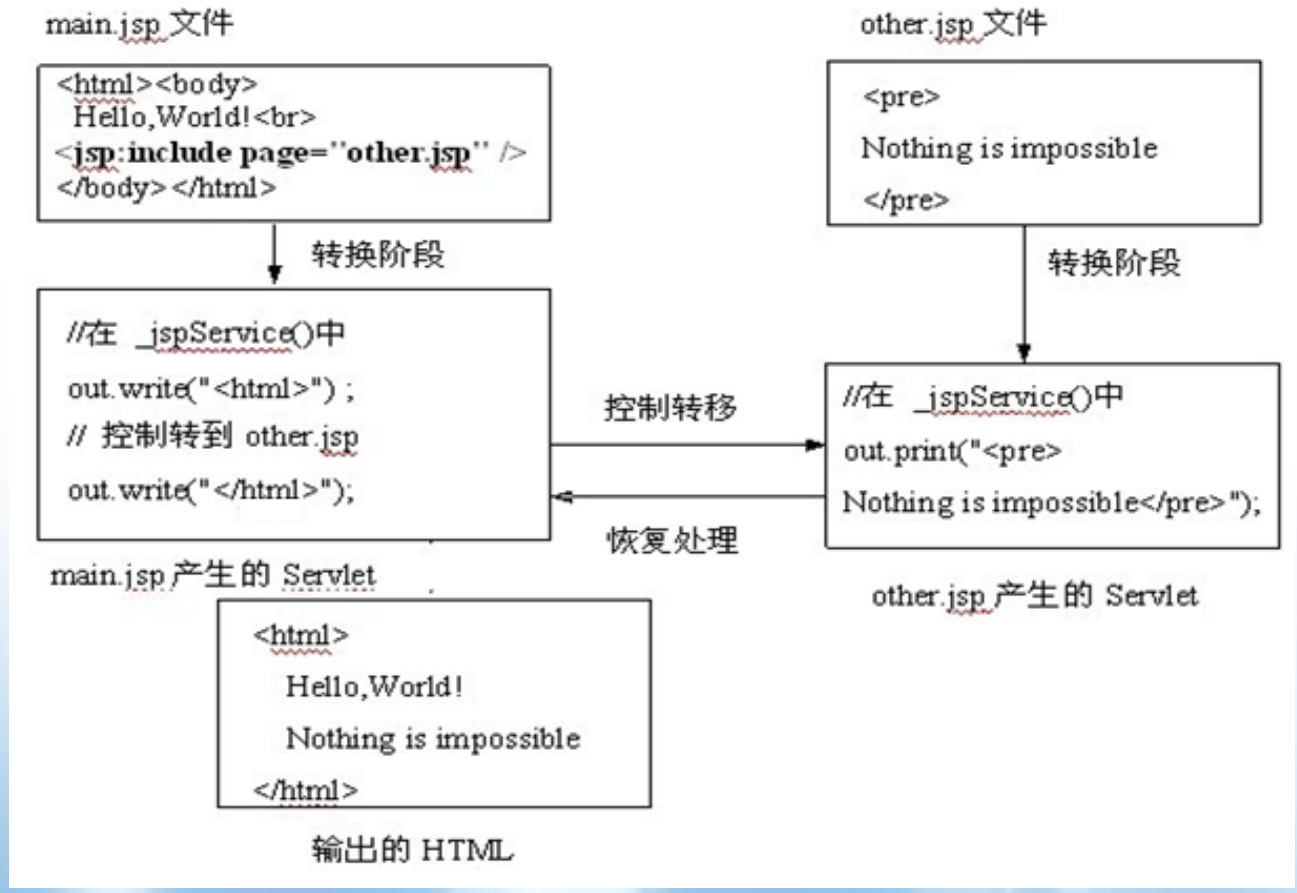
① page属性是必须的，其值必须是相对URL，并指向任何静态或动态Web组件，包括JSP页面、Servlet等。

② 可选的flush属性是指在将控制转向被包含页面之前是否刷新主页面。flush属性的默认值为false。

3.7.2 动态包含：include动作

- page属性的值可以是请求时表达式，例如：

```
<%! String pageURL = "other.jsp"; %>
<jsp:include page="<%= pageURL %>" />
```



3.7.2 动态包含：include动作

- 在功能上<jsp:include>动作的语义与RequestDispatcher接口的include()的语义相同
- 下面三个结构是等价的。

【结构1】

```
<%
RequestDispatcher rd =
 request.getRequestDispatcher("other.jsp");
rd.include(request, response);
%>
```

【结构2】

```
<%
    pageContext.include("other.jsp");
%>
```

【结构3】

```
<jsp:include page=“other.jsp" flush="true"/>
```

<jsp:include>与include指令的比较

include指令	<jsp:include>
<%@ include file=" "%>	<jsp:include page=" " flush="true"/>
修改了被包含的文件后，需更新源文件	修改了被包含的文件后，不需更新源文件
包含外部页面的过程在原JSP页面被编译成Servlet时进行	包含外部页面的过程在运行时进行

本质区别

<jsp:include>与include指令的比较

one.jsp

```
<%!
String var1="China";
%>
```

two.jsp

```
<%!
String var2="America";
String var3="England";
%>
```

three.jsp

```
<%  int j=1;
if (j==1){
%>
<%@ include file="one.jsp"%>
<%
}else{
%>
<%@ include file="two.jsp"%>
<%}%>
<%=var1%>  <%=var2%>  <%=va
```

! 问:执行three.jsp会出什么结果?

a.编译错误

b.显示China America England ✓

? --若将one.jsp与two.jsp中的!去掉,则

编译错误：变量未定义

--若将two.jsp中的var2改为var1,则

编译错误：变量重复定义

<jsp:include>与include指令的比较

one.jsp

```
<%!
String var1="China";
%>
```

two.jsp

```
<%!
String var2="America";
String var3="England";
%>
```

three.jsp

```
<%
int j=1;
String includeFile="";
if (j==1){
  includeFile = "one.jsp";
}else{
  includeFile = "two.jsp";
}%>
<jsp:include page="<%=includeFile%>"/>
<%=var1%>
```

问:执行three.jsp会出什么结果?

a.编译错误 ✔ var1未定义

b.显示China

1. 使用<jsp:param>传递参数

- 在<jsp:include>动作中可以使用<jsp:param />向被包含的页面传递参数

- 向somePage.jsp页面传递两个参数：

```
<jsp:include page="somePage.jsp">
   <jsp:param name="name1" value="value1"/>
   <jsp:param name="name2" value="value2"/>
</jsp:include>
```

1. 使用<jsp:param>传递参数

- 在<jsp:include>元素中可以嵌入任意多的<jsp:param>元素。value的属性值也可以像下面这样使用请求时属性表达式来指定。

```
<jsp:include page="somePage.jsp">
    <jsp:param name="name1" value="<%= someExpr1 %>" />
    <jsp:param name="name2" value="<%= someExpr2 %>" />
</jsp:include>
```

1. 使用<jsp:param>传递参数

- 通过<jsp:param>动作传递的“名/值”对保存在request对象中并只能由被包含的组件使用，在被包含的页面中使用request隐含对象的getParameter()获得传递来的参数。
- 这些参数的作用域是被包含的页面，在被包含的组件完成处理后，容器将从request对象中清除这些参数。
- <jsp:forward>动作的使用方法与<jsp:include>动作一样。

2. 与动态包含的组件共享对象

- 被包含的页面是单独执行的，因此它们不能共享在主页面中定义的变量和方法。然而，它们处理的请求对象是相同的，因此可以共享属于请求作用域的对象。下面程序说明了这一点。
- 程序hello2.jsp输出结果与程序4.11相同，但它使用了动态包含而不是静态包含。

✓ 主页面hello2.jsp通过调用request.setAttribute()把userName对象添加到请求作用域中，

✓ 被包含的页面response2.jsp通过调用request.getAttribute()检索该对象并使用表达式输出。

程序response2.jsp

- ✓ 在hello2.jsp文件中的隐含变量request与response2.jsp文件中的隐含变量request是请求作用域内的同一个对象。
- ✓ 对<jsp:forward>动作可以使用相同的机制。
- ✓ 除request对象外，还可以使用隐含变量session和application在被包含的页面中共享对象，但它们的作用域是不同的。

3.7.3 使用<jsp:forward>动作

- 使用<jsp:forward>动作把请求转发到其他组件，然后由转发到的组件把响应发送给客户，该动作的格式为：

```
<jsp:forward page="relativeURL" />
```

- page属性的值为转发到的组件的相对URL，它可以使用请求时属性表达式。它与<jsp:include>动作的不同之处在于，当转发到的页面处理完输出后，并不将控制转回主页面。使用<jsp:forward>动作，主页面也不能包含任何输出。

3.7.3 使用<jsp:forward>动作

- 在功能上<jsp:forward>的语义与RequestDispatcher接口的forward()的语义相同，因此下面三个结构是等价的。

【结构1】

```
<%
   RequestDispatcher rd =
request.getRequestDispatcher("other.jsp");
   rd.forward(request, response);
%>
```

3.7.3 使用<jsp:forward>动作

【结构2】

```
<%
    pageContext.forward("other.jsp");
%>
```

【结构3】

```
<jsp:forward page="other.jsp" />
```

- 在JSP页面中使用<jsp:forward>标准动作实际上实现的是控制逻辑的转移。
- 在MVC体系结构中，控制逻辑应该由控制器（Servlet）实现而不应该由视图（JSP页面）实现。因此，尽可能不在JSP页面中使用<jsp:forward>动作转发请求。

3.7.4 实例：使用包含设计页面布局

- Web应用程序界面应该具有统一的视觉效果，所有的页面都有同样的整体布局。一种比较典型的布局通常包含标题部分、脚注部分、菜单、广告区和主体实际内容部分。
- 设计这些页面时如果在所有的页面中都复制相同的代码，这不仅不符合模块化设计原则，将来若修改布局也非常麻烦。
- 使用JSP技术提供的include指令（<%@ include…）包含静态文件和include动作（<jsp:include …>）包含动态资源就可以实现一致的页面布局。

3.7.4 实例：使用包含设计页面布局

- 程序index.jsp页面使用 标签和include指令实现页面布局。

1、首面index.jsp

```
<%@ page contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>站点首页面</title>
<link href="css\layout.css" rel="stylesheet" type="text/css" />
</head>
<body>
     <div id="container">
        <div id="header"><%@ include file="header.jsp"%></div><br>
        <div id="topmenu"><%@ include file="topmenu.jsp"%></div>
        <div id="mainContent">
            <div id="leftmenu"><%@ include file="leftmenu.jsp"%></div>
            <div id="content"><%@ include file="content.jsp"%></div>
        </div>
        <div id="footer"><%@ include file="footer.jsp"%></div>
     </div>
</body>
</html>
```

2、标题页面header.jsp

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<script language="JavaScript" type="text/javascript">
function check() {
   open("register.jsp", "register");
}
</script>
<p>
<img alt="Here is a logo." src="images/head.jpg" />
</p>
<form action="login.do" method="post" name="login">
<p>
用户名<input type="text" name="username" size="13" />
密 码 <input type="password" name="password" size="13" />
     <input type="submit" name="submit" value="登  录">
     <input type="button" name="register" value="注  册" onclick="check();">
</form>
```

3、菜单页面topmenu.jsp

```
<%@ page contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<table border='0' align="left">
<tr >
    <td><a href="index.jsp">首页 | </a></td>
    <td><a href="DisplayServlet?category=101">手机数码  </a>|</td>
    <td><a href="DisplayServlet?category=102">家用电器</a>|</td>
    <td><a href="DisplayServlet?category=103">汽车用品</a>|</td>
    <td><a href="DisplayServlet?category=101">服饰鞋帽</a>|</td>
    <td><a href="DisplayServlet?category=101">运动健康</a>|</td>
    <td><a href="MyOrderServlet">我的订单</a>|</td>
    <td><a href="ShowCartServlet">查看购物车</a></td>
</tr></table>
```

4、左侧菜单页面leftmenu.jsp

```
<%@ page contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<br><p align="center"><b>商品分类</b></p>
<ul>
    <li><a href="DisplayServlet?category=101">手机数码</a></li>
    <li><a href="DisplayServlet?category=102">家用电器</a></li>
    <li><a href="DisplayServlet?category=103">汽车用品</a></li>
    <li><a href="DisplayServlet?category=101">服饰鞋帽</a></li>
    <li><a href="DisplayServlet?category=101">运动健康</a></li>
</ul>
```

5、主体内容页面content.jsp

```
<%@ page contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<table border="0">
   <tr><td colspan="2" align="center"><font color="blue">
   <b>圣诞惊喜！买华为笔记本，赢取华为平板</b></font></td>
  </tr>
  <tr><td width=50%><img src="images/phone.jpg">
       <p style="text-indent:2em">
      华为P30领取手机节优惠券，立减100元！再送：200元移动手机卡！
      派派价：3868元</p>
       <img src="images/gw.jpg"> </td>
   <td  width=50%><img src="images/comp.jpg">
      <p style="text-indent:2em">
          联想X280 12英寸笔记本电脑（i5-8400M 8G 1T 512M独显， Win10)
特价：3999元！</p>
      <img src="images/gw.jpg"></td>
  </tr>
</table>
```

6、页脚页面footer.jsp

```
<%@ page contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<hr />
<p align="center">关于我们|联系我们|人才招聘|友情链接</p>
<p align="center" ><font color="blue">
    Copyright &copy; 2020杭州派派电子商务有限责任公司,Tel:0571-
88991234</font>
</p>
```

3.7.4 实例：使用包含设计页面布局

- 在被包含的文件中，没有使用<html>和<body>等标签。实际上，它们不是完整的页面，而是页面片段
- 被包含文件名也可以完全不使用.jsp作为扩展名，而可以使用任何的扩展名，如.htmlf或.jspf等。
- 由于被包含的文件是由服务器访问的，因此可以将被包含的文件存放到Web应用程序的WEB-INF目录中，这样可以防止用户直接访问被包含的文件。

小　结

- JSP页面中有ServletContext、HttpSession、HttpServletRequest和PageContext等四个作用域对象
- Java Web开发中可以有多种方式重用Web组件。在JSP页面中包含组件的内容或输出实现Web组件的重用。有两种实现方式：使用include指令的静态包含和使用<jsp:include>动作的动态包含。