

Web应用开发 之 JSP标签技术

赵小敏

浙江工业大学计算机科学与技术学院

6.4 JSP标准标签库

➤ JSP标准动作:

- `<jsp:include page="content.jsp" />`
- `<jsp:useBean id="customer"`
- `class="com.demo.Customer" scope="session">`

➤ JSTL (JSP Standard Tag Library) 称为JSP标准标签库，是为实现Web应用程序常用功能而开发的标签库。

- `<c:set var="message"`
`value="世界那么大，我想去看看" scope="session" />`

6.4 JSP标准标签库

- 从JSP1.1版开始就可以在JSP页面中使用标签，使用标签不但可以实现代码重用，而且可以使JSP代码更简洁。
- 在JSP页面中可使用三种标签：
 - 标准动作标签（<jsp:include>、<jsp:useBean>等）
 - JSTL：JSP标准标签库（JSP Standard Tag Library。
 - 用户自定义的标签。

6.4 JSP标准标签库



- 6.4.1 JSTL概述
- 6.4.2 JSTL核心标签库
- 6.4.3 通用目的标签
- 6.4.4 条件控制标签
- 6.4.5 循环控制标签
- 6.4.6 URL相关的标签

6.4.1 JSTL概述

- 在使用JSTL前，应该获得JSTL包，并安装到Web应用程序中。可以到Apache Tomcat网站下载JSTL包，地址为<http://tomcat.apache.org/download-taglibs.cgi>。
- JSTL目前最新版本是1.2.5。
- 需下载两个文件，将它们复制到应用程序的WEB-INF/lib目录中。
 - [taglibs-standard-impl-1.2.5.jar](#)
 - [taglibs-standard-spec-1.2.5.jar](#)

6.4.1 JSTL概述

- 在Tomcat 9.0 安装的webapps\examples\WEB-INF\lib 目录中就包含 taglibs-standard-impl-1.2.5.jar 和 taglibs-standard-spec-1.2.5.jar两个文件，将它们复制到你的Web应用的WEB-INF/lib目录中即可。

apache-tomcat-9.0.19 > webapps > examples > WEB-INF > lib	
名称	修改日期
 taglibs-standard-impl-1.2.5.jar	2019/4/12 15:24
 taglibs-standard-spec-1.2.5.jar	2019/4/12 15:24

6.4.1 JSTL概述

- JSTL共提供了5个库，每个子库提供了一组实现特定功能的标签：
 - 核心标签库，包括通用处理的标签。
 - XML标签库，包括解析、查询和转换XML数据的标签。
 - 国际化和格式化库，包括国际化和格式化的标签。
 - SQL标签库，包括访问关系数据库的标签。
 - 函数库，包括管理String和集合的函数。

6.4.1 JSTL概述

库名称	使用的URI	前缀
核心标签库	http://java.sun.com/jsp/jstl/core	c
XML标签库	http://java.sun.com/jsp/jstl/xml	x
国际化和格式化库	http://java.sun.com/jsp/jstl/fmt	fmt
SQL标签库	http://java.sun.com/jsp/jstl/sql	sql
函数库	http://java.sun.com/jsp/jstl/functions	fn

6.4.2 JSTL核心标签库

- 核心（**core**）标签库可以分成4类。：
- 通用目的
 - `<c:out>`
 - `<c:set>`
 - `<c:remove>`
 - `<c:catch>`
- 循环控制
 - `<c:forEach>`
 - `<c:forTokens>`
- 条件控制
 - `<c:if>`
 - `<c:choose>`
 - `<c:when>`
 - `<c:otherwise>`
- URL处理
 - `<c:url>`
 - `<c:import>`
 - `<c:redirect>`
 - `<c:param>`

6.4.2 JSTL核心标签库

- 在JSP页面中使用JSTL，必须使用taglib指令来引用标签库
- 要使用核心标签库，必须在JSP页面中使用下面的taglib指令：

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

6.4.3 通用目的的标签

- 通用目的的标签包括
 - `<c:out>`
 - `<c:set>`
 - `<c:remove>`
 - `<c:catch>`

1. <c:out>标签

- <c:out>标签使用很简单，它有两种语法格式。

【格式1】不带标签体的情况

```
<c:out value = "value" [escapeXml="{true|false}"]  
      default = "defaultValue" />
```

- 如果escapeXml的值为true（默认值），表示将value属性值中包含的<、>、'、"或&等特殊字符转换为相应的实体引用（或字符编码），如小于号（<）将转换为<，大于号（>）将转换为>。如果escapeXml的值为false将不转换。

1. <c:out>标签

【格式2】带标签体的情况

```
<c:out value = "value" [escapeXml="{true|false}"]>
```

default value

```
</c:out>
```

在【格式2】中默认值是在标签体中给出的。

1. <c:out>标签

- 在value属性的值中可以使用EL表达式，例如：

```
<c:out value="${pageContext.request.remoteAddr}" />
```

```
<c:out value="${number}" />
```

- 上述代码分别输出客户地址和number变量的值。
- 从<c:out>标签的功能可以看到，它可以替换JSP的脚本表达式。

2. <c:set>标签

- <c:set>标签设置作用域变量以及对象（如JavaBeans与Map）的属性值。
- 该标签有下面4种语法格式。

【格式1】不带标签体的情况

```
<c:set var = "varName" value= "value"
```

```
[scope = "{page| request| session| application}"] />
```

2. <c:set>标签

【格式2】带标签体的情况

```
<c:set var = "varName" [scope =  
    "{page|request|session|application}"]>  
    body content  
</c:set>
```

【格式2】是在标签体中指定变量值。

2. <c:set>标签

- 下面两个标签：

```
<c:set var="number" value="${4*4}" scope="session" />
```

与

```
<c:set var="number" scope="session">
```

```
    ${4*4}
```

```
</c:set>
```

- 都将变量number的值设置为16，且其作用域为会话作用域。

2. <c:set>标签

- 使用<c:set>标签还可以设置指定对象的属性值，对象可以是JavaBeans或Map对象。
- 可以使用下面两种格式实现。

【格式3】不带标签体的情况

```
<c:set target = "target"  
        property = "propertyName"  
        value = "value" />
```

2. <c:set>标签

- 【格式4】带标签体的情况

```
<c:set target = "target"  
        property = "propertyName" >  
    body content  
</c:set>
```

- **target**属性指定对象名，**property**属性指定对象的属性名（JavaBeans的属性或Map的键）。
- 与设置变量值一样，属性值可以通过**value**属性或标签体内容指定。

2. <c:set>标签

- 程序setDemo.jsp为一个名为product的JavaBeans对象设置pname属性值。

```
<c:set target="${product}" property="pname" value="华为  
P40手机" />  
<c:out value="${sessionScope.product.pname}" />  
<br>  
<c:set target="${product}" property="pname">  
    联想笔记本电脑  
</c:set>  
<c:out value="${product.pname}" />
```

3. <c:remove>标签

- <c:remove>标签用来从作用域中删除变量，它的语法格式为：

<c:remove var="varName"

[scope = "{page| request| session| application}"] />

- var属性指定要删除的变量名，可选的scope属性指定作用域。
- 如果没有指定scope属性，容器将先在page作用域查找变量，然后是request，接下来是session，最后是application作用域，找到后将变量清除。

4. <c:catch>标签

- <c:catch>标签的功能是捕获标签体中出现的异常，语法格式为：

<c:catch [var = "*varName*"]>

body content

</c:catch>

- **var**是为捕获到的异常定义的变量名，当标签体中代码发生异常时，将由该变量引用异常对象，变量具有**page**作用域。

4. <c:catch>标签

示例：用 <c:catch>标签捕获异常

```
<c:catch var="myexception">
```

```
<%
```

```
    int i = 0;
```

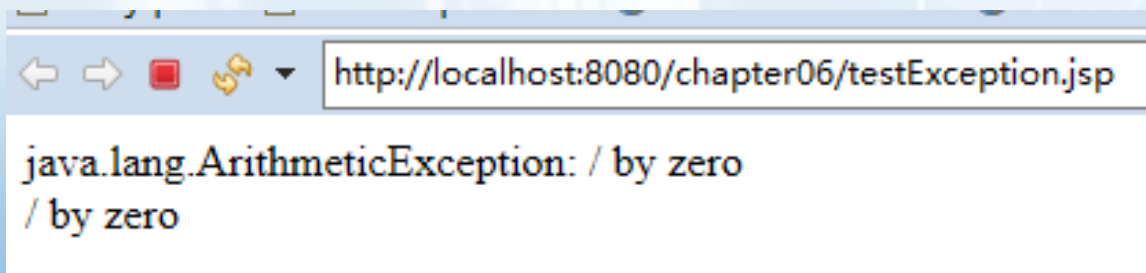
```
    int j = 10 / i; // 该语句发生异常
```

```
%>
```

```
</c:catch>
```

```
<c:out value="${myexception}" /><br>
```

```
<c:out value="${myexception.message}" />
```



6.4.4 条件控制标签

- 条件控制标签有4个：
 - `<c:if>`
 - `<c:choose>`
 - `<c:when>`
 - `<c:otherwise>`
- `<c:if>`和`<c:choose>`标签的功能类似于Java语言的if语句和switch-case语句。

1. <c:if>标签

- <c:if>标签用来进行条件判断，它有以下两种语法格式。

【格式1】不带标签体的情况

```
<c:if test="testCondition" var="varName"  
[scope = "{page| request| session| application}"] />
```

【格式2】带标签体的情况

```
<c:if test="testCondition" var="varName"  
[scope = "{page| request| session| application}"] >  
body content  
</c:if>
```

1. <c:if>标签

- 每个<c:if>标签必须有一个名为test的属性，它是一个boolean表达式。对于【格式1】，只将test的结果存于变量varName中。对于【格式2】，若test的结果为true，则执行标签体。
- 在下面代码中如果number的值等于16，则会显示其值。

```
<c:set var="number" value="${4*4}"  
  scope="session" />  
<c:if test="${number == 16}" var="result"  
  scope="session">  
  ${number}<br>  
</c:if> <br>  
<c:out value="${result}" />
```

2. `<c:choose>` 标签

- `<c:choose>` 标签类似于Java语言的switch-case语句，它本身不带任何属性，但包含多个`<c:when>`标签和一个`<c:otherwise>`标签，这些标签能够完成多分支结构。

2. <c:choose>标签

示例：根据color变量的值显示不同的文本

```
<c:set var="color" value="white" scope="session" />
```

```
<c:choose>
```

```
  <c:when test="${color == 'white'}">
```

白色!

```
  </c:when>
```

```
  <c:when test="${color == 'black'}">
```

黑色!

```
  </c:when>
```

```
  <c:otherwise>
```

其他颜色!

```
  </c:otherwise>
```

```
</c:choose>
```

6.4.5 循环控制标签

- 核心标签库的< c:forEach>和< c:forEachTokens>标签允许重复处理标签体内容。使用这些标签，能以三种方式控制循环的次数。
 - 对数的范围使用< c:forEach>以及它的begin、end和step属性。
 - 对Java集合中元素使用< c:forEach>以及它的var和items属性。
 - 对String对象中的令牌（token）使用< c:forEachTokens>以及它的items属性。

1. <c:forEach>标签

- <c:forEach>标签主要实现迭代，它可以对标签体迭代固定的次数，也可以在集合对象上迭代，该标签有两种格式。

【格式1】迭代固定的次数

```
<c:forEach [var="varName"] [begin="begin" end="end"
    step="step"] [varStatus="varStatusName"]>
```

body content

```
</c:forEach>
```

1. <c:forEach>标签

- <c:forEach>标签还可以嵌套，table99.jsp页面使用了嵌套的<c:forEach>标签实现输出九九表。

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core"
prefix="c" %>
<html><body>
<c:forEach var="x" begin="1" end="9" step="1">
    <c:forEach var="y" begin="1" end="{x}" step="1">
        ${y}*${x}=${x*y}
    </c:forEach>
    <br>
</c:forEach>
</body></html>
```

1. <c:forEach>标签

- 在<c:forEach>标签中还可以指定varStatus属性值来保存迭代的状态，例如，如果指定：

varStatus="status"

- 则可以通过**status**访问迭代的状态。其中包括：
本次迭代的索引、已经迭代的次数、是否是第一个迭代、是否是最后一个迭代等。它们分别用**status.index**、**status.count**、**status.first**、**status.last**访问。

1. <c:forEach>标签

- 程序foreach_1.jsp从0计数到10，每3个输出一个数。

```
<table border="1">
<th colspan="6">Example of forEach</th>
<tr><td>value of x</td>
    <td>status.index</td>
    <td>status.current</td>
    <td>status.count</td>
    <td>status.first</td>
    <td>status.last</td>
</tr>
<c:forEach var="x" varStatus="status" begin="10" end="20" step="3">
<tr><td align="center"><font color="blue">${x}</font></td>
    <td align="center">${status.index}</td>
    <td align="center">${status.current}</td>
    <td align="center">${status.count}</td>
    <td align="center">${status.first}</td>
    <td align="center">${status.last}</td>
</tr>
</c:forEach>
</table>
```

1. <c:forEach>标签

【格式2】在集合对象上迭代

```
<c:forEach var="varName" items="collection"  
[varStatus="statusName"][begin="begin" end="end"  
step="step"]>  
    body content  
</c:forEach>
```

- 这种迭代主要用于对Java集合对象的元素迭代，集合对象如List、Set或Map等。标签对每个元素处理一次标签体内容。这里，items属性值指定要迭代的集合对象，var用来指定一个作用域变量名，该变量只在<c:forEach>标签内部有效。

1. <c:forEach>标签

- 使用<c:forEach>标签显示List对象的元素。
假设有一个Book类定义如下。

```
package com.model;  
public class Book{  
    private String isbn;  
    private String title;  
    private double price;
```

```
// 这里省略了属性的setter方法和getter方法  
}
```

1. <c:forEach>标签

- BooksServlet创建一个List<Book>对象，然后将控制转发到books.jsp页面，在该页面中使用<c:forEach>标签访问每本书的信息。
- 在books.jsp页面中使用<c:forEach>标签访问列表中的元素，代码如下。

2. <c:forToken>标签

- 该标签用来在字符串中的令牌（**token**）上迭代，它的语法格式为：

```
<c:forTokens items="stringOfTokens" delims="delimiters"  
  [var="varName"]  
  [varStatus="varStatusName"]  
  [begin="begin"] [end="end"] [step="step"]>  
body content  
</c:forTokens>
```

2. <c:forToken>标签

- tokens.jsp使用<forTokens>标签输出一个字符串中各令牌的内容。

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"
%>
<html><body>
<c:set var="poems" value="白日依山尽,黄河入海流,欲穷千里目,更上
一层楼" />
<h4>登鹳雀楼 王之涣 </h4>
<c:forTokens var="line" items="${poems}" delims=", ">
    ${line}<br>
</c:forTokens>
</body></html>
```

6.4.6 URL相关的标签

- 与URL相关的标签有4个：
 - `<c:import>`
 - `<c:url>`
 - `<c:redirect>`
 - `<c:param>`

1. <c:param>标签

- <c:param>标签主要用于在<c:import>、<c:url>和<c:redirect>标签中指定请求参数，它的格式有下面两种。

【格式1】 参数值使用value属性指定

```
<c:param name="name" value="value" />
```

【格式2】 参数值在标签体中指定

```
<c:param name="name" >
```

```
param value
```

```
</c:param>
```


2. <c:import>标签

- <c:import>标签的功能与<jsp:include>标准动作的功能类似，可以将一个静态或动态资源包含到当前页面中。
- <c:import>标签有下面两种语法格式：

【格式1】资源内容作为字符串对象包含

```
<c:import url = "url" [context = "context"] [var =  
    "varName"]  
[scope = "{page| request| session| application}"]  
    [charEncoding = "charEncoding"]>  
    body content  
</c:import>
```

2. <c:import>标签

【格式2】资源内容作为Reader对象包含

```
<c:import url = "url" [context = "context"]
```

```
  [varReader = "varreaderName"]
```

```
  [charEncoding = "charEncoding"]
```

```
    body content
```

```
</c:import>
```

- **varReader**用于表示读取的文件的内容。其他属性与上面格式中含义相同。

2. <c:import>标签

- 程序importDemo.jsp使用<c:import>标签包含了footer.jsp页面，并向其传递了一个名为email的请求参数。
- 程序 footer.jsp是被包含的页面代码

3. <c:redirect>标签

- <c:redirect>标签的功能是将用户的请求重定向到另一个资源，它有两种语法格式。

【格式1】不带标签体的情况

```
<c:redirect url = "url" [context = "context"] />
```

【格式2】在标签体中指定查询参数

```
<c:redirect url = "url" [context = "context"] >
```

```
    <c:param> subtags
```

```
</c:redirect>
```

3. <c:redirect>标签

- 该标签的功能与 `HttpServletResponse` 的 `sendRedirect()` 的功能相同。它向客户发送一个重定向响应并告诉客户访问由 `url` 属性指定的 URL。
- 与 `<c:import>` 标签一样，可以使用 `context` 属性指定 URL 的上下文，也可以使用 `<c:param>` 标签添加请求参数。

3. <c:redirect>标签

- 下面的代码片段给出了一个<c:redirect>标签如何转向到一个新的URL的例子。

```
<c:redirect url="/content.jsp">
```

```
    <c:param name="par1" value="val1"/>
```

```
    <c:param name="par2" value="val2"/>
```

```
</c:redirect>
```

4. <c:url>标签

- 如果用户浏览器不接受Cookie，那么就需要重写URL来维护会话状态。
- 核心库提供了<c:url>标签。通过value属性来指定一个基URL，而转换的URL由JspWriter显示出来或者保存到由可选的var属性命名的变量中。

4. <c:url>标签

- <c:url>标签有如下两种格式。

【格式1】 不带标签体的情况

```
<c:url value="value" [context = "context"] [var="varName"]  
[scope="{page|request|session|application}"] />
```

- **value**属性指定需要重写的URL，**var**指定的变量存放URL值，**scope**属性来指定**var**的作用域。
- 如：<c:url value="/page.jsp" var="pagename"/>
- 由于**value**参数以斜杠开头，容器将把上下文名（假设为/chapter06）插入到该URL前面。

4. <c:url>标签

【格式2】带标签体的情况

```
<c:url value="value" [context = "context"] [var="varName"]  
[scope="{page|request|session|application}"] >  
    <c:param name="name" value="value">  
</c:url>
```

4. <c:url>标签

- 可以在<c:url>的标签中使用<c:param>标签向URL传递请求参数。下面代码给出了实现方法。

```
<c:url value="/page.jsp" var="pagename">  
  <c:param name="param1" value="${2*2}"/>  
  <c:param name="param2" value="${3*3}"/>  
</c:url>
```

- 在<c:param>标签中的参数通过name和value属性指定。如果浏览器接受Cookie，var属性的值将为：
`/chapter06/page.jsp?param1=4¶m2=9`

6.5 小 结

- JSTL是为实现Web应用程序常用功能而开发的标签库，它是由一些专家和用户开发的。使用JSTL可以提高JSP页面的开发效率，也可以避免重复开发标签库。
- JSTL由许多子库组成，每个子库提供了一组实现特定功能的标签，具体来说，这些子库包括core库、xml库、fmt库、sql库、functions库。

练习

1、把下面哪个代码放入简单标签的标签体中不可能输出9?

()

A. `${3+3+3}`

B. `"9"`

C. `<c:out value="9">`

D. `<%=27/3>`

2、下面哪个与`<%= var %>`产生的结果相同? ()

A. `<c:set value=var>`

B. `<c:var out=${var}>`

C. `<c:out value=${var}>`

D. `<c:out var="var">`

E. `<c:expr value=var>`

练习

3、下面代码的输出结果为 ()

```
<c:set value="3" var="a" />
```

```
<c:set value="5" var="b" />
```

```
<c:set value="7" var="c" />
```

```
${a div b}+${b mod c}
```

A. 5.6

B. 0.6+5

C. $a \div b + b \bmod c$

D. $3 \div 5 + 5 \bmod 7$

练习

为下面各段代码添入合法的属性名或标签名。

① `<c:forEach var="movie" items="{movieList}" _____ ="foo">`
 `${movie}`
 `</c:forEach>`

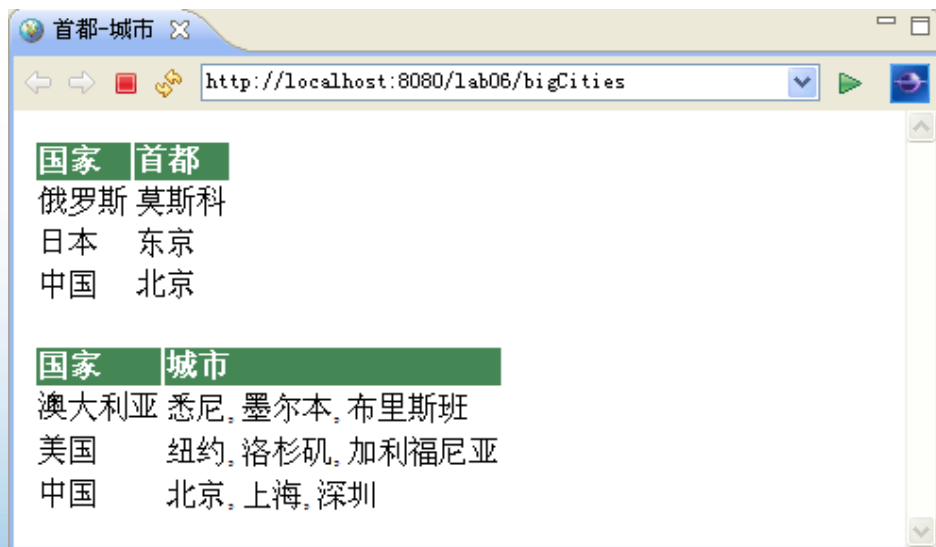
② `<c:if _____ ="${userPref == 'safety'}" >`
 Mybe you should just walk...
 `</c:if>`

③ `<c:set var="userLevel" scope="session" _____ ="foo" />`

④ `<c:choose>`
 `<c: _____ ="${userPref == 'performance'}">`
 Now you can stop even if you `do` drive insanely fast.
 `</c: _____ >`
 `<c: _____ >`
 Our brakes are the best.
 `</c: _____ >`
 `</c:choose>`

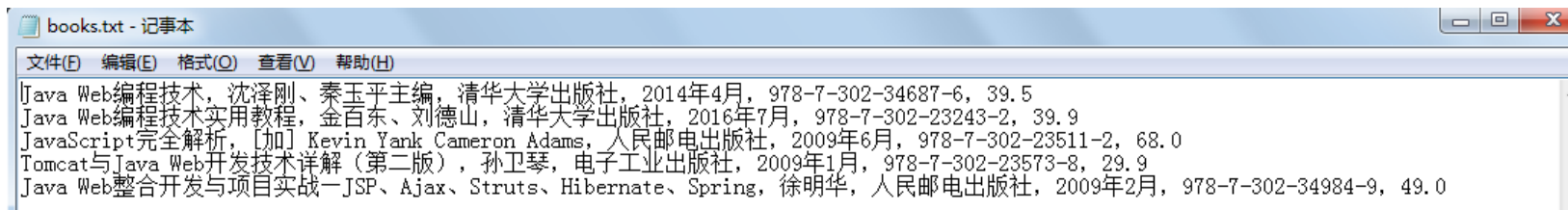
作业

1、编写BigCitiesServlet类和bigCities.jsp页面实现<c:forEach>标签对Map对象迭代。在BigCitiesServlet类中创建一个Map<String,String>对象capitals，键为国家名称，值为首都名称，添加几个对象（不少于三个国家和首都）。另外创建一个Map<String,String[]>对象bigCities，键为国家名称，值为String数组包含该国家的几个大城市。在doGet()中使用RequestDispatcher对象将请求转发到bigCities.jsp页面，在bigCities.jsp页面中显示这些国家的首都和城市，效果如下图所示：



作业

2、用JSP标签和MVC模式设计实现如下功能：输入书名、作者或ISBN号可以模糊查询出图书信息，以列表形式显示，其中图书信息存在book.txt文件或数据库中，包括书名、作者、出版社、出版时间、ISBN号、价格等，具体信息如下：



附图书的信息：

Java Web编程技术, 沈泽刚、秦玉平主编, 清华大学出版社, 2014年4月, 978-7-302-34687-6, 39.5

Java Web编程技术实用教程, 金百东、刘德山, 清华大学出版社, 2016年7月, 978-7-302-23243-2, 39.9

JavaScript完全解析, [加] Kevin Yank Cameron Adams, 人民邮电出版社, 2009年6月, 978-7-302-23511-2, 68.0

Tomcat与Java Web开发技术详解(第二版), 孙卫琴, 电子工业出版社, 2009年1月, 978-7-302-23573-8, 29.9

Java Web整合开发与项目实战—JSP、Ajax、Struts、Hibernate、Spring, 徐明华, 人民邮电出版社, 2009年2月, 978-7-302-34984-9, 49.0