# 第 5 章

# JSTL 标签库

JSTL(Jakarta Server Pages Standard Tag Library)是 JSP 标准标签库的简称。

在早期的 JSP 开发中,JSP 与 Servlet 的职责定位模糊,开发人员习惯上在 JSP 中做很多的逻辑工作,HTML 代码与 Java 代码混杂,逻辑功能、控制功能、视图功能都混淆在了一起。这种开发模式给代码的开发、阅读、维护都带来很大的麻烦。

随着 MVC 架构的出现,Jakarta EE 中增加了 JSTL 标准。JSTL 的目标是提供给 Jakarta Web 开发人员一个标准的、通用的标签库,开发人员可以利用这些标签取代 JSP 页面上的 Java 代码,从而提高程序的可读性,降低程序的维护难度。

JSTL 在本质上是提前定义好的一组标签,这些标签封装了不同的功能,在页面上调用 JSTL,可以大幅减少 JSP 文件中的 Java 代码。这使 Java 代码与 HTML 代码明显分离,因此使用 JSTL 标签库更符合 MVC 设计理念。使用 JSTL 后,JSP 可以专注于视图功能,为 Jakarta Web 开发带来非常大的好处。

Eclipse 集成 Tomcat 进行 Jakarta Web 开发时,需要单独导入 JSTL 相关包。下载 jstl-impl-3.0.jar 和 jstl-api-3.0.jar,复制到 WEB-INF/lib 文件夹下即可使用 JSTL。

JSTL 3.0 包含的标签库内容见表 5-1。

表 5-1　JSTL 3.0 包含的标签库内容

| 功　能 | URI | 前　缀 |
|---|---|---|
| 核心库 | jakarta.tags.core | c |
| XML 处理 | jakarta.tags.xml | x |
| I18N 格式化 | jakarta.tags.fmt | fmt |
| 关系型数据库访问 SQL | jakarta.tags.sql | sql |
| 函数 | jakarta.tags.functions | fn |

注意,Jakarta EE 10 中包含的 JSTL 是 3.0 版。Jakarta EE 10 对应的 JSP 版本是 3.1。JSTL 3.0 需要 JSP 3.1 及以上的 Web 容器支持。EL 表达式是 JSP 规范的一部分,在 JSTL 操作中会大量使用 EL,而 EL 是从 JSP 2.0 规范开始的。

本章只着重介绍几个常用标签的使用,其他内容参见 Jakarta JSTL 规范。


视频讲解

## 5.1 自定义标签库

标签库由标签库描述文件 tld(tag-library descriptors)和标签实现类两部分组成。JSTL 标签库中的所有标签与自定义标签定义方法相同。下面自定义一个作者标签,显示作者的名字和性别,通过这个案例了解 JSTL 标签库的定义过程。

(1) 定义标签类。

新建 Web 项目 ETCTag,创建包 com.icss.tag,定义标签类 AuthorTag 继承 Jakarta.servlet.jsp.tagext.TagSupport。一般会重写 doEndTag()方法。

```
public class AuthorTag extends TagSupport{
    private String name;
    private String sex;
    public String getSex() {
        return sex;
    }
    public void setSex(String sex) {
        this.sex = sex;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int doEndTag() throws JspException {
        JspWriter out = pageContext.getOut();
            try{
                out.println("<table bgColor=yellow>");
                out.println("<tr>");
                if(sex != null)
                  out.println("<td>作者:" + name
                            + ",性别: " + sex + ",welcome you!!!</td>");
                else
                  out.println("<td>作者:" + name
                              + ",welcome you!!!</td>");
                out.println("</tr>");
                out.println("</table>");
                }catch(Exception ex){
                  ex.printStackTrace();
                }
                return this.EVAL_PAGE;
    }
}
```

（2）定义标签库描述文件 etc.tld。

<taglib>的子元素<short-name>为标签库前缀。一个<taglib>下可以定义多个标签<tag>。

标签<tag>的子元素<name>为标签名，<tag-class>为标签实现类，<attribute>为标签的属性。

<attribute>属性的子元素<name>是属性名，<required>表示属性是否为必须设置，false 表示可选，true 表示必须设置值；<rtexprvalue>表示是否可以动态赋值，false 表示只能设置静态值，true 表示可以动态赋值，如可以使用 Java 表达式、EL 表达式或<jsp:attribute>赋值。

```
<taglib>
    <description>etc our library</description>
    <display-name>etc library</display-name>
    <tlib-version>3.0</tlib-version>
    <short-name>etc</short-name>
    <uri>http://com.icss.tag/core</uri>
  <tag>
    <description>
    show file author
    </description>
    <name>author</name>
    <tag-class>com.icss.tag.AuthorTag</tag-class>
    <body-content>JSP</body-content>
    <attribute>
        <description>
          author name
        </description>
        <name>name</name>
        <required>true</required>
        <rtexprvalue>true</rtexprvalue>
    </attribute>
    <attribute>
        <description>
          author sex
        </description>
        <name>sex</name>
        <required>false</required>
        <rtexprvalue>true</rtexprvalue>
    </attribute>
  </tag>
</taglib>
```

（3）在 JSP 项目 TagUse 中调用自定义标签。

- 把 AuthorTag 编译成 lib 文件，导入 JSP 项目中。
- 把 etc.tld 复制到 JSP 项目的 WEB-INF 下。
- 在 JSP 文件中调用自定义标签。

```
<%@ page language="java" import="java.util.*" pageEncoding="GBK"%>
<%@ taglib uri="http://com.icss.tag/core" prefix="etc"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
    <body>
        This is author tag use sample page. <br>
        <etc:author name="xiaohp"></etc:author>
        <br/>
      <etc:author name="xiaohp" sex="男"></etc:author>
    </body>
</html>
```

## 5.2 核心标签库

核心标签库前缀统一使用<c:>标识，这个标签库作用最大、用途最广。

参见 jstl-impl-3.0.jar/META-INF/c.tld 中核心标签描述：

```
<taglib>
    <description>JSTL core library</description>
    <display-name>JSTL core</display-name>
    <tlib-version>3.0</tlib-version>
    <short-name>c</short-name>
    <uri>jakarta.tags.core</uri>
</taglib>
```

使用标签库时，需要在 JSP 页面的头部用 taglib 指令声明，<short-name>表示标签前缀，<uri>在标签声明时需要指明。

```
<%@ taglib prefix="c" uri="jakarta.tags.core"%>
```


视频讲解

### 5.2.1 一般用途标签

本节介绍几个一般用途的常用标签，分别为<c:out>、<c:set>、<c:remove>。<c:catch>标签不常用，本节不作介绍。

#### 1. <c:out>

<c:out>标签与<%=脚本表达式 %> 或 ${el 表达式}的功能基本一致，用于在页面输出静态或动态信息。

基本语法如下，属性信息见表 5-2。

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

表 5-2 <c:out>属性信息

| 属 性 名 | 动态支持 | 属性类型 | 描 述 |
|---|---|---|---|
| value | true | Object | 要输出的表达式值 |
| escapeXml | true | Boolean | 决定字符<,>、&、',"在结果串中是否需要转义成对应的字符实体代码,默认为 true。HTML 转义符见表 1-4 |
| default | true | Object | 输出结果为 null 时,显示默认值 |

### 示例 5-1

```
<p>
    你有 <c:out value="${sessionScope.user.itemCount}"/> 个选项
</p>
```

### 示例 5-2

当 EL 表达式输出为 null 时,可以使用<c:out>输出默认值。

所属城市:

```
<c:out value="${customer.address.city}" default="未知"/>
```

### 2. <c:set>

基本语法如下,属性信息见表 5-3。

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

表 5-3 <c:set>属性信息

| 属 性 名 | 动 态 支 持 | 属 性 类 型 | 描 述 |
|---|---|---|---|
| value | true | Object | 要设置的表达式值 |
| var | true | String | 存储在域对象中的变量名 |
| scope | false | String | 与 var 对应的域对象 |

### 示例 5-3

使用 Java 脚本计算一个随机数,然后用<c:set>标签把随机数存储到 request 域对象中,最后用 EL 表达式提取 request 对象中的变量 r,用<c:out>输出随机数。

```
<body>
    <%
```

```
            int rand = (int)(Math.random() * 100);
        %>
        <c:set value = "<% = rand %>" var = "r" scope = "request" />
        获取随机数: <c:out value = "${r}"></c:out>
    </body>
```

### 3. <c:remove>

基本语法如下,属性信息见表 5-4。

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

表 5-4 <c:remove>属性信息

| 属 性 名 | 动 态 支 持 | 属 性 类 型 | 描 述 |
|---|---|---|---|
| var | false | String | 存储在域对象中的变量名 |
| scope | false | String | 与 var 对应的域对象 |

**示例 5-4**

使用<c:remove>移除 request 域中存储的随机数 rand。

```
<body>
    <%
        int rand = (int)(Math.random() * 100);
        request.setAttribute("rand", rand);
    %>
    获取随机数: <c:out value = "${rand}"></c:out>
    <br>
    <c:remove var = "rand" scope = "request"/>
    移除后随机数: <c:out value = "${rand}"></c:out>
</body>
```

视频讲解

## 5.2.2 条件判断标签

Java 脚本中,条件判断语句有 if…else、if…else if…else if…else 等。JSTL 的目的是简化 Java 脚本,因此条件判断标签必不可少。

**示例 5-5**

```
<c:if test = "${user.visitCount == 1}">
    This is your first visit. Welcome to the site!
</c:if>
```

### 1. <c:if>

条件判断标签，如果测试条件为 true，显示标签体内容。标签定义如下：

```
<tag>
    <name>if</name>
    <tag-class>org.apache.taglibs.standard.tag.rt.core.IfTag</tag-class>
    <body-content>JSP</body-content>
    <attribute>
        <name>test</name>
        <required>true</required>
        <rtexprvalue>true</rtexprvalue>
        <type>boolean</type>
    </attribute>
    <attribute>
        <name>var</name>
        <required>false</required>
        <rtexprvalue>false</rtexprvalue>
    </attribute>
    <attribute>
        <name>scope</name>
        <required>false</required>
        <rtexprvalue>false</rtexprvalue>
    </attribute>
</tag>
```

基本语法如下，属性信息见表 5-5，test 为必选属性，var 和 scope 为可选属性。

```
<c:if test="测试条件"
    [var="varName"] [scope="{page|request|session|application}"]>
        内容体
</c:if>
```

表 5-5 <c:if>属性信息

| 属 性 名 | 动 态 支 持 | 属 性 类 型 | 描 述 |
|---|---|---|---|
| test | true | Boolean | 测试条件返回为真，处理内容体信息，否则不处理 |
| var | false | String | 把测试条件返回的布尔值结果，存储到 scope 对象中 |
| scope | false | String | page、request、session、application 中的某个域对象 |

**示例 5-6**

JSTL 与 EL 标签配合使用，判断图书价格是否小于或等于 100(符合预算)，如是，则显示图书名称。

```
<c:if test="${book.price <= 100}">
    The book ${book.title} fits your budget!
</c:if>
```

**示例 5-7**

测试请求参数中是否有一个名字为 name 的参数，如果参数为空，则提示。

```
<c:if test="${empty param.name}">
    Please specify your name.
</c:if>
```

**示例 5-8**

判断随机数是否大于 50，并用 EL 表达式输出测试结果。

```
<body>
    <%
        int rand = (int)(Math.random() * 100);
        pageContext.setAttribute("rand", rand);
    %>
    <c:if test="${rand>50}" var="aa" scope="request">
            welcome you!
    </c:if>
    测试结果为: ${aa}
</body>
```

**总结**

JSTL 中只有<c:if>判断，没有 else 判断。如果需要 if…else 判断结构，可以用<c:if>标签判断两次，如<c:if test="${user==null}">和<c:if test="${user!=null}">同时使用。

### 2. <c:choose><c:when><c:otherwise>

标签<c:choose><c:when><c:otherwise>配合使用，执行效果等同于 Java 脚本中的条件判断语句。

(1) <c:choose>基本语法。

```
<c:choose>
    内容体 (<when> 和 <otherwise> 子标签)
</c:choose>
```

<c:choose>中可以放 1～n 个<c:when>标签，即至少要包含一个子标签<c:when>，可以同时存在多个<c:when>子标签。

<c:choose>中可以放 0～1 个<c:otherwise>子标签，即可以没有<c:otherwise>子标签，最多只能有一个<c:otherwise>子标签。

（2）<c:when>基本语法。

```
<c:when test="测试条件">
    内容体
</c:when>
```

test 属性返回 boolean 值，决定内容体是否被处理。

<c:when>不能单独使用，必须要有<c:choose>父标签。

<c:when>必须要出现在<c:otherwise>标签的前面。

（3）<c:otherwise>基本语法。

```
<c:otherwise>
    条件块
</c:otherwise>
```

在<c:choose>标签体内，如果没有<c:when>就返回真，则 JSP 容器处理<c:otherwise>中的条件块。

<c:otherwise>不能单独使用，必须要有<c:choose>父标签。

<c:otherwise>在<c:choose>体内，必须出现在最后。

**示例 5-9**

示例效果等同于 if…else if…else if…else，只要任何一个 test 返回真，就不再向下判断，执行内容体后，立即跳出<c:choose>。

```
<c:choose>
    <c:when test="${user.category == 'trial'}">
        …
    </c:when>
    <c:when test="${user.category == 'member'}">
        …
    </c:when>
    <c:when test="${user.category == 'vip'}">
        …
    </c:when>
    <c:otherwise>
        …
    </c:otherwise>
</c:choose>
```

**示例 5-10**

示例效果等同于 if…else。

```
<c:choose>
    <c:when test="${count == 0}">
```

```
            No records matched your selection.
        </c:when>
        <c:otherwise>
            ${count} records matched your selection.
        </c:otherwise>
    </c:choose>
```


视频讲解

## 5.2.3 迭代标签<c:forEach>

标签<c:forEach>在JSP页面迭代输出集合中的数据，与Java脚本中的for()循环功能类似，基本语法如下，属性信息见表5-6。


视频讲解

```
<c:forEach [var="varName"]  items="collection"
           [varStatus="varStatusName"]
           [begin="begin"] [end="end"] [step="step"]>
    内容体
</c:forEach>
```

表5-6 <c:forEach>属性信息

| 属性名 | 动态支持 | 属性类型 | 描述 |
|---|---|---|---|
| var | false | String | 迭代变量名 |
| items | true | 参见集合说明 | 迭代的集合对象 |
| varStatus | false | String | 显示迭代状态，参见LoopTagStatus接口 |
| begin | true | int | 迭代操作时，集合的起始索引 |
| end | true | int | 迭代操作时，集合的结束索引 |
| step | true | int | 迭代操作的步长值 |

<c:forEach>说明信息：

- begin索引必须大于或等于0。
- 如果end指定的索引值小于begin，则迭代不会执行。
- 步长值step必须大于或等于1。
- 如果items集合为null，不会抛出异常，按照空集合处理，即迭代不会执行。

items支持如下集合类型：

- 基本类型的静态数组(迭代时会自动使用包装类)。
- java.util.Collection接口的实现类(会调用iterator()方法迭代集合)。
- java.util.Iterator接口的实现类。
- java.util.Enumeration接口的实现类。
- java.util.Map接口实现类，var变量的类型为java.util.Map.Entry。
- 通用分隔符隔开的一个字符串。

### 1. 集合迭代

迭代静态数组、java.util.Collection、java.util.Iterator等集合对象。

**示例 5-11**

迭代输出产品集合中的产品信息。

```
<c:forEach var="product" items="${products}">
    产品名：${product.name} 价格：${product.price}
</c:forEach>
```

**示例 5-12**

迭代输出顾客信息，并在<table>中显示。

```
<table>
    <c:forEach var="ct" items="${customers}">
        <tr><td>顾客：</td><td>${ct.name}</td></tr>
    </c:forEach>
</table>
```

### 2. Map 迭代

当 items 类型为 java.util.Map 时，每个 item 的类型是 java.util.Map.Entry。它有两个属性：key 和 value。

**示例 5-13**

```
<c:forEach var="entry" items="${myHashtable}">
    元素 key 是：      ${entry.key}   <br>
    元素 value 是：    ${entry.value}
</c:forEach>
```

**示例 5-14**

如下代码中的${entry.value}表示 User 对象，${entry.value.uname}表示用户名。

```
<body>
    <%
        Map map = new HashMap<String,User>();
        map.put("tom", new User("tom"));
        map.put("jack", new User("jack"));
        map.put("rose", new User("rose"));
        request.setAttribute("map",map);
    %>
    <table>
        <c:forEach var="entry" items="${map}">
            <tr>
                <td>${entry.key}</td>
```

```
                <td>${entry.value.uname}</td>
            </tr>
        </c:forEach>
    </table>
</body>
```

**示例 5-15**

使用嵌套<c:forEach>读取 Map 数据，${aParam.value}是集合对象。

```
<c:forEach var="aParam" items="${paramValues}">
    参数名：${aParam.key}
    参数值:
    <c:forEach var="aValue" items="${aParam.value}">
        ${aValue}
    </c:forEach>
    <br>
</c:forEach>
```

### 3. 迭代状态

使用 varStatus 显示当前迭代状态。迭代状态的操作依赖接口 jakarta.servlet.jsp.jstl.core.LoopTagStatus，接口定义如下：

```
public interface LoopTagStatus {
    java.lang.Integer    getBegin();
    int                  getCount();
    java.lang.Object     getCurrent();
    java.lang.Integer    getEnd();
    int                  getIndex();
    java.lang.Integer    getStep();
    boolean              isFirst();
    boolean              isLast();
}
```

- getBegin：返回迭代的 begin 属性值，如果 begin 属性不存在，就返回 null。
- getCount：返回环绕迭代集合的当前数量。count 是个相对值，从 1 开始。例如迭代某个集合，begin=5，end=15，step=5，则 counts 值分别为 1、2、3。
- getCurrent：返回当前正在迭代的对象。
- getEnd：返回迭代的 end 属性值，如果 end 属性不存在，则返回 null。
- getIndex：返回环绕迭代集合的当前索引值，索引从 0 开始。
- getStep：返回迭代的 step 属性值，如果 step 属性不存在，则返回 null。
- isFirst：如果当前的迭代项是集合的第一项，则返回 true。
- isLast：如何当前的迭代项是集合的最后一项，则返回 true。

**示例 5-16**

```
    <body>
    <%
        int[] sz = new int[]{10,12,35,24,65};
        request.setAttribute("sz",sz);
    %>
    <table>
        <c:forEach var="item" items="${sz}" varStatus="st">
            <tr>
                <td>${st.count}</td>
                <td>${st.index}</td>
                <c:if test="${st.index % 2 == 0}">
                    <td>${item}</td>
                </c:if>
            </tr>
        </c:forEach>
    </table>
</body>
```

输出结果如下：

```
1    0    10
2    1
3    2    35
4    3
5    4    65
```

**4. 范围属性**

使用属性 begin、end、step，可以在迭代集合时选择部分内容处理。

**示例 5-17**

```
<body>
    <%
        int[] sz = new int[]{10,12,35,24,65,78,102,205,12,31,20,309};
        request.setAttribute("sz",sz);
    %>
    <table>
        <c:forEach var="item" items="${sz}" varStatus="st"
              begin="2" end="10" step="2" >
            <tr>
                <td>${st.count}</td>
                <td>${st.index}</td>
                <td>${item}</td>
            </tr>
```

```
        </c:forEach>
    </table>
</body>
```

输出结果如下：

```
1    2     35
2    4     65
3    6     102
4    8     12
5    10    20
```

### 5.2.4 URL 相关标签

在 JSP 页面链接、导入、重定向到其他 URL 资源，在 JSTL 中统称为 URL 相关标签。

#### 1. <c:url>

<c:url>基本语法如下，属性信息见表 5-7。

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

表 5-7 <c:url>属性信息

| 属 性 名 | 动 态 支 持 | 属 性 类 型 | 描 述 |
|---|---|---|---|
| value | true | String | 要处理的 URL |
| context | true | String | 相对路径的 URL 依赖的外部 Context 名字 |
| var | false | String | 输出到 scope 对象的变量名 |
| scope | false | String | 域对象 |

<c:url>与锚点<a href="#">不同，它用于生成访问资源的 URL，但是并不会产生超链接。value 属性值可以使用绝对地址，也可使用相对地址。绝对地址不会重写，而相对地址在解析时会重写，会自动增加 context 前缀。如<c:url value="/ads/logo.html"/>，假设 Web 站点 context 为：/foo，则输出结果：/foo/ads/logo.html。

**示例 5-18**

重写<img>图片的相对路径，当前站点是/Hello，则图片链接为/*Hello*/*pic*/*flex*.*png*。

```
<body>
    <img src="${pageContext.request.contextPath}/pic/flex.png">
    等效于
    <img src="<%=request.getContextPath()%>/pic/flex.png">
    等效于
    <img src="<c:url value='/pic/flex.png'/>" />
```

```
    <a href = "<c:url value = '/pic/flex.png'/>"> flex 图片</a>
</body>
```

**示例 5-19**

设置 URL 资源后，将它存储在域对象中，然后通过 EL 表达式读取。

```
<body>
    <c:url value = "/pic/flex.png" var = "flexUrl" scope = "page"></c:url>
    <img src = "${flexUrl}" />
</body>
```

**示例 5-20**

url 设置 context 属性后，强制指向外部 context，图片指向/*Center*/*pic*/*flex*.*png*。

```
<body>
    <c:url value = "/pic/flex.png" context = "/Center" var = "flexUrl"></c:url>
    <img src = "${flexUrl}" />
</body>
```

### 2. <c:import>

<c:import>用于导入基于 URL 的资源，基本语法如下，属性信息见表 5-8。

```
<c:import url = "url"    [context = "context"]
      [var = "varName"] [scope = "{page|request|session|application}"]
      [charEncoding = "charEncoding"]>
  可选内容体 <c:param> 子标签
</c:import>
```

表 5-8 <c:import>属性信息

| 属 性 名 | 动态支持 | 属性类型 | 描 述 |
|---|---|---|---|
| url | true | String | 待导入资源的 URL，可以是相对路径，也可以是绝对路径 |
| context | true | String | 当使用相对路径访问外部 context 资源时，指明外部 context 的名字 |
| var | false | String | 用于存储所引入文本的变量 |
| scope | false | String | var 属性存储在哪个域对象 |
| charEncoding | true | String | 引入资源的字符编码 |

说明：

- 如果 url 为 null、空串或无效，JspException 异常被抛出；
- 如果 charEncoding 为 null 或空串，这个配置会被忽略；
- <c:import>与<jsp:include>的功能相似，但是<jsp:include>只能嵌入当前 Web 应用的资源，而<c:import>还可以导入外部资源。

**示例 5-21**

嵌入头文件。

(1) 编写头文件 head.jsp。

```
<img src="${pageContext.request.contextPath}/pic/flex.png">
welcome ${user.uname}
```

(2) 在 hello.jsp 中嵌入 head.jsp。

```
<body>
    <c:import url="/main/head.jsp"></c:import>
    hello!
</body>
```

**示例 5-22**

使用绝对地址,导入外部资源文件。

```
<c:import url="ftp://ftp.acme.com/README"/>
```

### 3. <c:redirect>

<c:redirect>表示发送 HTTP 的 redirect 到客户端,与调用 HttpServletResponse.sendRedirect()相同,基本语法如下,属性见表 5-9。

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

表 5-9 <c:redirect>属性信息

| 属性名 | 动态支持 | 属性类型 | 描 述 |
|---|---|---|---|
| url | true | String | 重定向的目标 URL,绝对地址和相对地址都可以 |
| context | true | String | 当使用相对路径访问外部 context 资源时,指明外部 context 的名字 |

**示例 5-23**

用户登录后,根据身份不同,跳转到不同的页面。

```
<body>
    <c:choose>
        <c:when test="${user.role==1}">
            <c:redirect url="/main/back.jsp" />
        </c:when>
        <c:when test="${user.role==2}">
            <c:redirect url="/main/hello.jsp" />
        </c:when>
```

```
        <c:otherwise>
            <c:redirect url="http://www.sina.com.cn" />
        </c:otherwise>
    </c:choose>
</body>
```

#### 4. <c:param>

<c:param>表示添加请求参数给 URL，在<c:import>、<c:url>、<c:redirect>中都可以使用，基本语法如下，属性信息见表 5-10。

```
<c:param name="参数名" value="参数值"/>
```

表 5-10 <c:param>属性信息

| 属 性 名 | 动 态 支 持 | 属 性 类 型 | 描 述 |
|---|---|---|---|
| name | true | String | HTTP 请求参数名 |
| value | true | String | HTTP 请求参数值 |

**示例 5-24**

```
<c:import url="/exec/doIt">
    <c:param name="action" value="register"/>
</c:import>
```

使用<c:param>与 URL 中直接绑定参数的效果相同：

```
<c:import url="/exec/doIt?action=register"/>
```

## 5.3 格式化标签库

在 JSP 页面显示数字、货币、百分比、日期、时间等数据时，需要按照本地敏感或定制格式显示，这时使用格式化标签库非常方便。

### 5.3.1 格式化数字、货币、百分比

<fmt:formatNumber>标签可以格式化数字、货币、百分比，基本语法如下：

```
<fmt:formatNumber value="数值"
                  [type="{number|currency|percent}"]
                  [pattern="定制格式"]
                  [currencyCode="货币编码"]
```

```
        [currencySymbol = "货币符号"]
        [groupingUsed = "{true|false}"]
        [maxIntegerDigits = "最大整数数字"]
        [minIntegerDigits = "最小整数数字"]
        [maxFractionDigits = "最大小数数字"]
        [minFractionDigits = "最小小数数字"]
        [var = "变量名"]
        [scope = "{page|request|session|application}"]/>
```

### 示例 5-25

```
<%@ taglib prefix = "fmt" uri = "jakarta.tags.fmt" %>
<!DOCTYPE html>
<html>
    <body>
        <fmt:formatNumber value = "9876543.21" type = "currency"/>
    </body>
</html>
```

输出结果(自动按本地的货币格式显示)：￥9,876,543.21。

如果设置本地化环境为 en_US,则输出结果为美国货币格式：$9,876,543.21。

```
<body>
    <fmt:setLocale value = "en_US" />
    <fmt:formatNumber value = "9876543.21" type = "currency"/>
</body>
```

### 示例 5-26

```
<body>
    <%
        int rand = (int)(Math.random() * 100);
        request.setAttribute("rand", rand);
     %>
    <fmt:formatNumber value = "${rand/3}" pattern = ".00"/>
    <fmt:formatNumber value = "12.3" pattern = ".00"/>
</body>
```

输出结果(保留两位小数)：

```
25.33
12.30
```

### 5.3.2 格式化日期和时间

使用< fmt:formatDate >标签可以在 JSP 页面格式化日期和时间，基本语法如下：

```
< fmt:formatDate  value = "日期"
                  [type = "{time|date|both}"]
                  [dateStyle = "{default|short|medium|long|full}"]
                  [timeStyle = "{default|short|medium|long|full}"]
                  [pattern = "定制格式"]
                  [timeZone = "时区"]
                  [var = "变量名"]
                  [scope = "{page|request|session|application}"]/>
```

**示例 5-27**

```
<%@ taglib prefix = "fmt" uri = "jakarta.tags.fmt" %>
<!DOCTYPE html>
<html>
    <body>
        <fmt:formatDate value = "<% = new Date() %>" pattern = "dd/MM/yyyy"/> <br>
        <fmt:formatDate value = "<% = new Date() %>" pattern = "yyyy - MM - dd HH:mm"/>
    </body>
</html>
```

输出结果：

```
28/02/2020
2020 - 02 - 28  20:46
```

## 5.4 本章习题

(1) 下列不属于核心库标签的是(　　)。

A. choose　　B. if　　C. fmt　　D. otherwise

(2) 下面不属于 JSTL 标签的是(　　)。

A. 核心标签库　　B. 国际化/格式化标签库

C. HTML 标签库　　D. SQL 标签库

E. 函数标签库

(3) 以下代码配置了一个自定义标签，这段代码应该位于(　　)。

```
<description>Spring Framework JSP Tag Library</description>
<tlib-version>4.0</tlib-version>
<short-name>spring</short-name>
<uri>http://www.springframework.org/tags</uri>
```

A. Tomcat 的 conf/server.xml 文件中

B. Jakarta Web 应用的 WEB-INF/web.xml 文件中

C. 一个 tld 文件中

D. Jakarta Web 应用的 META-INF/context.xml 文件中

(4) 以下不属于<c:forEach>标签属性的是(　　)。

A. var　　B. value　　C. items　　D. varStatus

(5) 如下数据类型中,<c:forEach>标签不能迭代的是(　　)。

A. java.util.Collection　　B. 静态数组

C. java.util.Map　　D. org.w3c.dom.Node

E. java.util.Iterator

(6) 以下不属于<c:out>标签属性的是(　　)。

A. var　　B. value　　C. escapeXml　　D. default

(7) <c:out>标签与<%=脚本表达式 %> 或 ${el 表达式}的功能基本一致。　(对/错)

(8) 使用<c:set>标签,可以把指定的值存储到 pageContext 域对象中。　(对/错)

(9) <c:choose>中至少要包含一个子标签<c:when>和一个 <c:otherwise>。　(对/错)

(10) <c:import>导入资源,url 可以是绝对路径也可以是相对路径。　(对/错)