JSTL

JSTL简介

JSTL全称：JSP Standard Tag Library（JSP标准标签库），是一个一段完善的开源的JSP标签库。 作用：替代jsp的代码脚本。 优点：让jsp页面变得更简洁。

标签的引入

功能范围	URI	前缀
核心标签库–重点	http://java.sun.com/jsp/jstl/core	c
格式化	http://java.sun.com/jsp/jstl/fmt	fmt
函数	http://java.sun.com/jsp/jstl/functions	fn
数据库(不使用)	http://java.sun.com/jsp/jstl/sql	sql
XML(不使用)	http://java.sun.com/jsp/jstl/xml	x

为了在 JSP 页面使用 JSTL 类库，必须以下列格式使用 taglib 指令：

```
CORE 标签库
<%@ taglib prefix=“c” uri=“http://java.sun.com/jsp/jstl/core” %>
XML 标签库
<%@ taglib prefix=“x” uri=“http://java.sun.com/jsp/jstl/xml” %>
FMT 标签库
```

```
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
SQL 标签库
<%@ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql" %>
FUNCTIONS 标签库
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>
```

前缀可以是任意内容，遵循规范可以使团队中由不同人员编写的代码更加相似；所以，建议使用事先设计好的前缀。

导入依赖

```
<dependency>
     <groupId>javax.servlet</groupId>
     <artifactId>javax.servlet-api</artifactId>
     <version>3.1.0</version>
     <scope>provided</scope>
   </dependency>
   <dependency>
     <groupId>org.apache.taglibs</groupId>
     <artifactId>taglibs-standard-impl</artifactId>
     <version>1.2.5</version>
   </dependency>
   <dependency>
     <groupId>javax.servlet.jsp.jstl</groupId>
     <artifactId>jstl-api</artifactId>
     <version>1.2</version>
   </dependency>
```

但会出现找不到标签库的错误，舍弃上面的依赖

1. 把jstl-1.2.jar和standard.jar两个jar包拷贝到tomcat目录的lib目录下，因为tomcat没有这两个jar包
2. 然后引入jstl1.0jar包中的c.tld、c-1_0.tld、c-1_0-rt.tld文件在WEB-INF文件夹中。

条件动作标签

条件动作指令用于处理页面的输出结果依赖于某些输入值的情况，在 Java 中是利用 if、if…else 和switch 语句来进行处理的。在 JSTL 中也有 4 个标签可以执行条件式动作指令：if、 choose、when 和otherwise。

if标签

if标签先对某个条件进行测试，如果该条件运算结果为true,则处理它的主体内容，测试结果保存在一个Boolean对象中，并创建一个限域变量来引用Boolean对象。可以利用var属性设置限域变量名，利用scope属性来指定其最小作用范围。

注：if标签没有else 需要多个if来实现

```
<%--
<c:if />
    if 标签用来做 if 判断。
    test 属性表示判断的条件（使用 EL 表达式输出）
--%>
<c:if test="${ 12 == 12 }">
    <h1>12 等于 12</h1>
</c:if>
<c:if test="${ 12 != 12 }">
    <h1>12 不等于 12</h1>
</c:if>
```

choose、when、otherwise标签

通常这三个标签放在一起来使用。

- <c:choose>标签嵌套在 <c:when>和 <c:otherwise>标签的外面作为父标签来使用。
- <c:choose>标签和 <c:when>标签也可以组合使用。

```
<%--
    <c:choose> <c:when> <c:otherwise>标签
    作用：多路判断。跟 switch ... case .... default 非常接近

    choose 标签开始选择判断
    when 标签表示每一种判断情况
    test 属性表示当前这种判断情况的值
    otherwise 标签表示剩下的情况

    <c:choose> <c:when> <c:otherwise>标签使用时需要注意的点：
    1、标签里不能使用 html 注释，要使用 jsp 注释
    2、when 标签的父标签一定要是 choose 标签
--%>
<%
    request.setAttribute("height", 180);
%>
<c:choose>
    <%-- 这是 html 注释 --%>
    <c:when test="${ requestScope.height > 190 }">
        <h2>小巨人</h2>
```

```
    </c:when>
    <c:when test="${ requestScope.height > 180 }">
        <h2>很高</h2>
    </c:when>
    <c:when test="${ requestScope.height > 170 }">
        <h2>还可以</h2>
    </c:when>
    <c:otherwise>
        <c:choose>
            <c:when test="${requestScope.height > 160}">
                <h3>大于 160</h3>
            </c:when>
            <c:when test="${requestScope.height > 150}">
                <h3>大于 150</h3>
            </c:when>
            <c:when test="${requestScope.height > 140}">
                <h3>大于 140</h3>
            </c:when>
            <c:otherwise>
                其他小于 140
            </c:otherwise>
        </c:choose>
    </c:otherwise>
</c:choose>
```

forEach

根据循环条件遍历集合中的元素

- var设定变量名用于存储从集合中取出的元素（item）（必须，无默认值）
- items指定要遍历的集合
- begin、end用于指定遍历的起始位置和终止位置（有默认值）
- step指定循环的步长（有默认值 1）
- varStatus通过index(下标)、count（循环数）、first（是否第一次被循环）、last（是否最后一次被循环）几个状态值，描述begin和end子集中的元素的状态。

普通for

```
<c:forEach begin="1" end="10" var="item">
    <%--1. --%>
    ${item} <br>
</c:forEach>
```

增强for 遍历数组

```
<%--
2.遍历 Object 数组
    for (Object item: arr)
    items 表示遍历的数据源（遍历的集合）
    var 表示当前遍历到的数据
--%>
<%
    request.setAttribute("arr", new String[]
{"18610541354","18688886666","18699998888"});
%>
<c:forEach items="${ requestScope.arr }" var="item">
    ${ item } <br>
</c:forEach>
```

增强for 遍历 Map 集合

```
<%--遍历 Map 集合--%>
<%
    Map<String,Object> map = new HashMap<String, Object>();
    map.put("key1", "value1");
    map.put("key2", "value2");
    map.put("key3", "value3");
    // for ( Map.Entry<String,Object> entry : map.entrySet()) {
    // }
    request.setAttribute("map", map);
%>
<c:forEach items="${ requestScope.map }" var="entry">
    <%--输出所有的键值对--%>
    ${entry}
    <%--只输出我要的值--%>
    ${entry.key}
    ${entry.value}
    <%--按照自己希望的格式输出 key 和 value--%>
    <h1>${entry.key} = ${entry.value}</h1>
</c:forEach>
```

```
<%--4.遍历 List 集合---list 中存放 Student 类，有属性：编号，用户名，密码，年龄，电话信息--%>
<%
    List<String> studentList = new ArrayList<>();
    for (int i = 1; i <= 10; i++) {
        studentList.add("aoligei"+i);
    }
    request.setAttribute("stus", studentList);
%>
<table>
    <tr>
        <th>内容</th>
        <th>下标</th>
```

```
            <th>循环数</th>
            <th>第一次? </th>
            <th>最后一次? </th>
            <th>步长</th>
        </tr>
        <%--
            items 表示遍历的集合
            var 表示遍历到的数据
            begin 表示遍历的开始索引值
            end 表示结束的索引值
            step 属性表示遍历的步长值
            varStatus 属性表示当前遍历到的数据的状态
        for (int i = 1; i < 10; i+=2)
        --%>
        <c:forEach begin="2" end="7" step="2" varStatus="itemp"
items="${requestScope.stus}" var="stu">
            <tr>
                <td>${stu}</td>
                <td>${itemp.index}</td>
                <td>${itemp.count}</td>
                <td>${itemp.first}</td>
                <td>${itemp.last}</td>
                <td>${itemp.step}</td>
                <td>${itemp.begin}</td>
                <td>${itemp.end}</td>
            </tr>
        </c:forEach>
    </table>
```

格式化动作标签

formatNumber

我们这次先来看下formatNumber标签的使用。先了解下它的作用，fmt:formatNumber标签用于格式化数字，百分比，货币。

来看下语法格式：

属性	描述	是否必要	默认值
value	要显示的数字	是	无
type	NUMBER，CURRENCY，或PERCENT类型	否	Number

属性	描述	是否必要	默认值
pattern	指定一个自定义的格式化模式用与输出	否	无
currencyCode	货币码（当type="currency"时）	否	取决于默认区域
currencySymbol	货币符号 (当 type="currency"时)	否	取决于默认区域
groupingUsed	是否对数字分组 (TRUE 或 FALSE)	否	true
maxIntegerDigits	整型数最大的位数	否	无
minIntegerDigits	整型数最小的位数	否	无
maxFractionDigits	小数点后最大的位数	否	无
minFractionDigits	小数点后最小的位数	否	无
var	存储格式化数字的变量	否	Print to page
scope	var属性的作用域	否	page

如果type属性为percent或number，那么我们就可以使用其它几个格式化数字属性。maxIntegerDigits属性和minIntegerDigits属性允许我们指定整数的长度。若实际数字超过了maxIntegerDigits所指定的最大值，则数字将会被截断 。

有一些属性允许我们指定小数点后的位数。minFractionalDigits属性和maxFractionalDigits属性允许我们指定小数点后的位数。若实际的数字超出了所指定的范围，则这个数字会被截断。

数字分组可以用来在每三个数字中插入一个逗号。groupingIsUsed属性用来指定是否使用数字分组。当与minIntegerDigits属性一同使用时，就必须要很小心地来获取预期的结果了。

我们或许会使用pattern属性。这个属性可以让您在对数字编码时包含指定的字符。我们来具体看下这些字符：

符号	描述
0	代表一位数字
E	使用指数格式

符号	描述
#	代表一位数字，若没有则显示 0，前导 0 和追尾 0 不显示。
.	小数点
,	数字分组分隔符
;	分隔格式
-	使用默认负数前缀
%	百分数
?	千分数
¤	货币符号，使用实际的货币符号代替
X	指定可以作为前缀或后缀的字符
'	在前缀或后缀中引用特殊字符

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>

<html>
<head>
  <title>JSTL fmt:formatNumber 标签</title>
</head>
<body>
<h3>数字格式化:</h3>
<c:set var="balance" value="120000.2309" />
<fmt:formatNumber value="${balance}"
            type="number" var="num" scope="request"/></p> ${num}
<p>格式化数字 (1): <fmt:formatNumber value="${balance}"
            type="currency"/></p>
<p>格式化数字 (2): <fmt:formatNumber type="number"
            maxIntegerDigits="3" value="${balance}" /></p>
<p>格式化数字 (3): <fmt:formatNumber type="number"
            maxFractionDigits="3" value="${balance}" /></p>
<p>格式化数字 (4): <fmt:formatNumber type="number"
            groupingUsed="false" value="${balance}" /></p>
<p>格式化数字 (5): <fmt:formatNumber type="percent"
            maxIntegerDigits="3" value="${balance}" /></p>
<p>格式化数字 (6): <fmt:formatNumber type="percent"
            minFractionDigits="10" value="${balance}" /></p>
<p>格式化数字 (7): <fmt:formatNumber type="percent"
            maxIntegerDigits="3" value="${balance}" /></p>
<p>格式化数字 (8): <fmt:formatNumber type="number"
            pattern="###.###E0" value="${balance}" /></p>
<p>美元 :
<fmt:setLocale value="en_US"/>
```

```
<fmt:formatNumber value="${balance}" type="currency"/></p>
</body>
</html>
```

formatDate

fmt:formatDate标签具有以下属性：

属性	描述	必需	默认值
value	要显示的日期值	Yes	None
type	DATE, TIME, or BOTH	No	date
dateStyle	FULL, LONG, MEDIUM, SHORT, or DEFAULT	No	default
timeStyle	FULL, LONG, MEDIUM, SHORT, or DEFAULT	No	default
pattern	Custom formatting pattern	No	None
timeZone	Time zone of the displayed date	No	Default time zone
var	Name of the variable to store the formatted date	No	Print to page
scope	Scope of the variable to store the formatted date	No	page

pattern属性来指定更精确的处理日期：

Code	Purpose	Sample
G	The era designator	AD
y	The year	2002
M	The month	April & 04
d	The day of the month	20
h	The hour(12-hour time)	12

Code	Purpose	Sample
H	The hour(24-hour time)	0
m	The minute	45
s	The second	52
S	The millisecond	970
E	The day of the week	Tuesday
D	The day of the year	180
F	The day of the week in the month	2 (2nd Wed in month)
w	The week in the year	27
W	The week in the month	2
a	The a.m./p.m. indicator	PM
k	The hour(12-hour time)	24
K	The hour(24-hour time)	0
z	The time zone	Central Standard Time
'		The escape for text
''		The single quote

```
<%
    Date date=new Date();
    pageContext.setAttribute("now",date);
%>

<p>Formatted Date (1): <fmt:formatDate type="time"
            value="${now}" /></p>
<p>Formatted Date (2): <fmt:formatDate type="date"
            value="${now}" /></p>
<p>Formatted Date (3): <fmt:formatDate type="both"
            value="${now}" /></p>
<p>Formatted Date (4): <fmt:formatDate type="both"
            dateStyle="short" timeStyle="short"
            value="${now}" /></p>
<p>Formatted Date (5): <fmt:formatDate type="both"
            dateStyle="medium" timeStyle="medium"
            value="${now}" /></p>
<p>Formatted Date (6): <fmt:formatDate type="both"
            dateStyle="long" timeStyle="long"
            value="${now}" /></p>
<p>Formatted Date (7): <fmt:formatDate pattern="yyyy-MM-dd"
            value="${now}" /></p>
```

```
${pageScope.now}
<br>
<fmt:formatDate value="${pageScope.now}" pattern="yyyy-MM-dd hh:mm:ss"/>
```