# JSTL

## JSTL简介

* JSP虽然为我们提供了EL表达式用来替代JSP表达式，但是由于EL表达式仅仅具有输出功能，而不替代页面中的JSP脚本片段。
* 为了解决这个问题，JSP为我们提供了可以自定义标签库(Tag Library)的功能。
* 所谓自定义标签库就是指可以在JSP页面中以类似于HTML标签的形式调用Java中的方法。使用方法和我们JSP动作标签类似。
* 而为了方便开发使用Sun公司又定义了一套通用的标签库名为JSTL(JSP Standard Tag Library)，里面定义很多我们开发中常用的方法，方便我们使用。
* JSTL的标准由Sun公司定制，Apache的Jakarta小组负责实现。
* JSTL由5个不同功能的标签库组成。

## 使用JSTL

* 使用JSTL必须在项目中导入两个jar包
  + taglibs-standard-impl-1.2.1.jar
  + taglibs-standard-spec-1.2.5.jar
* 然后还需要在JSP页面中通过taglib标签引入标签库。
  + <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
  + prefix用来指定前缀名，我们通过该名来使用JSTL
  + uri相当于库的唯一标识，因为JSTL由多个不同的库组成，使用该属性指定要导入哪个库。
* 使用JSTL
  + <c:out value="hello"></c:out>
  + 这个例子标识，调用前缀为c的标签的out方法，向页面中输出value属性中的字符串。
  + JSTL的使用非常像html标签。

## 标签库

* JSTL由五个不同功能的标签库组成。

|  |  |  |
| --- | --- | --- |
| 功能范围 | URI | 前缀 |
| 核心 | 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 |

### 核心标签(Core Tags)

* Core标签库，包括了我们最常用的标签。
* 要使用Core标签库需要在JSP页面中加入：
  + <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

#### <c:out>

* <c:out>用于计算一个表达式并将结果输出到当前页面。
* 功能类似于JSP表达式<%= >和EL表达式${}
* 可以设置的属性
  + value
    - 作用：要输出的值
    - 参数类型：Object
  + default
    - 作用：当value为null时显示的默认值。
    - 参数类型：Object
  + escaXml
    - 作用：是否对特殊字符进行转义。
    - 参数类型：boolean
* 例：
  + <c:out value="${user.name}" default="" escapeXml="true"></c:out>

#### <c:set>

* <c:set>用于添加或修改域中的属性
* 可以设置的属性：
  + value
    - 作用：要设置的值
    - 参数类型：Object
  + var
    - 作用：表示域中存放的属性名。
    - 参数类型：String
  + scope
    - 作用：指定域（page、request、session、application），若不指定则为page域对象
    - 参数类型：String
  + target
    - 作用：要修改的域对象的属性名（必须是JavaBean或者Map）
    - 参数类型：Object
  + property
    - 作用：指定要修改的对象的属性名
    - 参数类型：String
* 例
  + 设置属性：
    - <c:set var="key" value="value" scope="request"></c:set>
  + 修改属性：
    - <c:set property="name" target="${user}" value="孙悟空"></c:set>

#### <c:remove>

* <c:remove>用于移除域中的属性
* 可以设置的属性：
  + var
    - 作用：设置要移除的属性的名字
    - 作用：String
  + scope
    - 作用：设置要移除属性所在的域，若不指定则删除所有域中的对应属性
    - 参数类型：String
* 例：
  + 移除所有域中key属性：<c:remove var="key"/>
  + 移除request中的key属性: <c:remove var="key" scope="request"/>

#### <c:if>

* <c:if>用于实现if语句的判断功能
* 可设置的属性
  + test
    - 作用：设置if判断的条件，用于判断标签体是否被执行
    - 参数类型：boolean
  + var
    - 作用：用于指定接收判断结果的变量名
    - 参数类型：boolean
  + scope
    - 作用：指定判断结果保存到哪个域
    - 参数类型：String
* 例：

<c:if test="${empty user }" var="isUserEmpty" scope="request">

用户为空

</c:if>

#### <c:choose>、<c:when>、<c:otherwise>

* <c:choose>、<c:when>、<c:otherwise>三个标签配合使用，功能类似于Java中的if/else
* <c:choose>是<c:when>和<c:otherwise>的父标签
* <c:when>的属性
  + test
    - 作用：用于设置判断条件，若正确则c:when中的代码执行，否则不执行
    - 参数类型：boolean
* <c:otherwise>
  + 作用：如果所有的<c:when>都没有执行则执行<c:otherwise>的标签体
* 例：

<c:choose>

<c:when test="${param.age>=18 }">

您已经成年

</c:when>

<c:otherwise>

您未成年

</c:otherwise>

</c:choose>

#### <c:forEach>

* <c:forEach>用于对多个对象的集合进行迭代，重复执行标签体，或者重复迭代固定的次数。
* 可设置属性：
  + var
    - 作用：设置遍历出对象的名称
    - 参数类型：String
  + items
    - 作用：指定要遍历的集合对象
    - 参数类型：数组、字符串和各种集合
  + varStatus
    - 作用：指定保存迭代状态的对象的名字，该变量引用的是一个LoopTagStatus类型的对象，通过该对象可以获得一些遍历的状态：
      * count
      * index
      * first
      * last
      * name
    - 参数类型：String
  + begin
    - 作用：指定遍历的开始位置
    - 参数类型：int
  + end
    - 作用：指定遍历的结束位置
    - 参数类型：int
  + step
    - 作用：迭代的步长
    - 参数类型：int
* 例：

<c:forEach items="${list}" var="user" begin="0" end="4" step="2" varStatus="vs">

${vs.index} -- ${user.name } -- ${user.age }<br />

</c:forEach>

#### <c:url>

* <c:url>主要用来重写URL地址
  + 可设置的属性：
    - value
      * 作用：设置要处理的URI地址，注意这里要以/开头
      * 可接受参数：String
    - var
      * 作用：修改后存储到域对象中的uri属性名。
      * 可接受参数：String
    - scope
      * 作用：设置修改后uri存放的域
      * 可接受参数：String
* 例：
  + 使用相对路径：

<c:url value="index.jsp" var="uri" scope="request">

<c:param name="name" value="张三"></c:param>

</c:url>

* + - 会生成如下地址：index.jsp?name=%E5%BC%A0%E4%B8%89
  + 使用绝对路径会自动在路径前加上项目名：

<c:url value="/index.jsp" var="uri" scope="request">

<c:param name="name" value="张三"></c:param>

</c:url>

* + - 会生成如下地址：/Test\_JSTL/index.jsp?name=%E5%BC%A0%E4%B8%89

#### <c:redirect>

* <c:redirect>主要用于将请求重定向到另一个资源地址
* 可设置的属性：
  + url
    - 作用：指定要重定向到的目标地址，注意这里指定绝对路径会自动加上项目名
    - 参数类型：String
* 例：

<c:redirect url="/target.jsp"></c:redirect>

### JSTL函数（JSTL Functions）

* 函数标签库是在JSTL中定义的标准的EL函数集。
* 函数标签库中定义的函数基本上都是对字符串的操作。
* 引入：<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>

#### fn:contains和fn:containsIgnoreCase

作用：用于判断字符串中是否包含指定字符串，containsIgnoreCase忽略大小写。

语法：fn:contains(string,subString) 🡪 boolean

参数和返回值：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 作用 |
| string | String | 源字符串 |
| subString | String | 要查找的字符串 |
| 返回值 | boolean | 若string中包含subString则返回true，否则返回false |

例：

${fn:contains("hello","HE")} 🡪 false

${fn:containsIgnoreCase("hello","HE")} 🡪 true

#### fn:startsWith和fn:endsWith

作用：判断一个字符串是否以指定字符开头（startsWith）或结尾（endsWith）

语法：fn:startsWith(string , prefix) 🡪 boolean

fn:endsWith(string,suffix) 🡪 boolean

参数和返回值：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 作用 |
| string | String | 源字符串 |
| prefix 或 suffix | String | 要查找的前缀或后缀字符串 |
| 返回值 | boolean | 符合要求返回true，否则返回false |

例：${fn:startsWith("hello","he") } 🡪 true

${fn:endsWith("hello","he") } 🡪 false

#### fn:indexOf

作用：在一个字符串中查找指定字符串，并返回第一个符合的字符串的第一个字符的索引。

语法：fn:indexOf(string,subString) 🡪 int

参数和返回值：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 作用 |
| string | String | 源字符串 |
| subString | String | 要查找的字符串 |
| 返回值 | int | 若在string中找到subString则返回第一个符合的索引，若没有符合的则返回-1 |

例：${fn:indexOf("hello",'e') } 🡪 1

#### fn:replace

作用：将一个字符串替换为另外一个字符串，并返回替换结果

语法：fn:replace(str , beforeSubString , afterSubString) 🡪 String

参数和返回值：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 作用 |
| str | String | 源字符串 |
| beforeSubString | String | 被替换的字符串 |
| afterSubString | String | 要替换的新字符串 |
| 返回值 | String | 替换后的字符串 |

例：${fn:replace("hello","llo",'e') } 🡪 hee

#### fn:substring

作用：截取字符串

语法：fn: substring (str , beginIndex , endIndex) 🡪 String

参数和返回值：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 作用 |
| str | String | 源字符串 |
| beginIndex | int | 开始位置索引(包含该位置) |
| endIndex | int | 结束位置索引(不包含自身) |
| 返回值 | String | 返回截取的字符串 |

例：${fn:substring("hello",1,3) } 🡪 el

#### fn:substringBefore和fn:substringAfter

作用：返回一个字符串指定子串之前（substringBefore）之后（substringAfter）的字符串

语法：fn: substringBefore(string,subString) 🡪 String

fn: substringAfter (string,subString) 🡪 String

参数和返回值：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 作用 |
| str | String | 源字符串 |
| subString | int | 指定str中的一个子串，该串之前或之后的字符串将被返回。 |
| 返回值 | String | 返回截取的字符串 |

例：${fn:substringBefore("hello","l") } 🡪 he

${fn:substringAfter("hello","l") } 🡪 lo

#### fn:split

作用：将一个字符串拆分成字符串数组。

语法：fn:split(string,delimiters) 🡪 String

参数和返回值：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 作用 |
| str | String | 要被拆分的字符串 |
| delimiters | String | 指定根据什么内容拆分字符串 |
| 返回值 | String[] | 返回拆分后的字符串数组 |

例：${fn:split("a-b-c-d-e-f-g","-")} 🡪 返回一个数组对象[a,b,c,d,e,f,g]

#### fn:join

作用：将数组中所有元素连接成一个字符串

语法：fn:join(array,sparator) 🡪 String

参数和返回值：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 作用 |
| str | String | 要被拆分的字符串 |
| sparator | String | 在结果中每个元素之间的分隔符 |
| 返回值 | String[] | 拼接之后的结果 |

例：

|  |
| --- |
| <%  String[] strs = new String[]{"a","b","c","d","e","f"};  pageContext.setAttribute("strs", strs);  %>  ${fn:join(strs,'-') } |

返回：a-b-c-d-e-f

#### fn:toLowerCase和fn:toUpperCase

作用：将字符串都转换成大写（toUpperCase）或小写（toLowerCase）字符

语法：

fn: toLowerCase (str) 🡪 String

fn: toUpperCase(str) 🡪 String

参数和返回值：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 作用 |
| str | String | 源字符串 |
| 返回值 | String | 转换为大写或小写的字符串 |

例：${fn:toLowerCase("ABCDEFG") } 🡪 abcdefg

${fn:toUpperCase("abcdefg") } 🡪 ABCDEFG

#### fn:trim

作用：去掉字符串的前后空格

用法：fn:trim(str) 🡪 String

参数和返回值：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 作用 |
| str | String | 源字符串 |
| 返回值 | String | 去掉前后空格的结果 |

例：${fn:trim(" hello ") } 🡪 hello

#### fn:length

作用：返回集合或者字符串的长度

用法：fn: length (str) 🡪 int

参数和返回值：

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 作用 |
| input | String、集合、数组 | 要计算长度的目标 |
| 返回值 | int | 集合或字符串的长度 |

例：${fn:length("hello") }🡪 5