**自定义标签：**

**我们可以自己去定义一些jsp标签。自定义包含两个：java类，tld文件(标签的描述文件)**

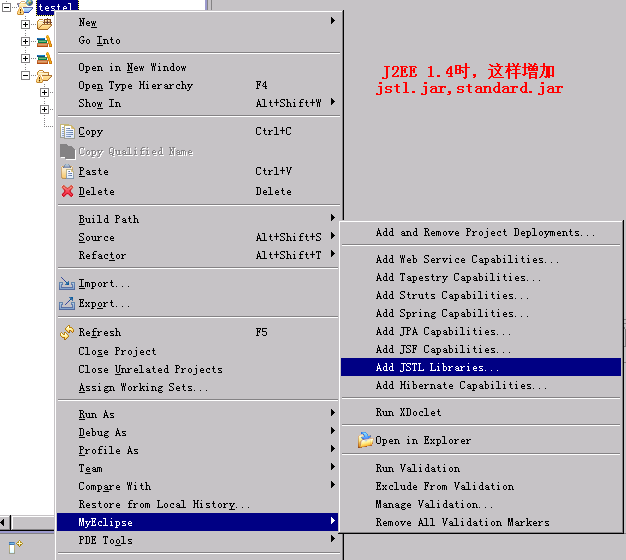
**标签库JSTL（JSP Standard Tag Library）：**

因为在jsp中使用java代码去控制页面的显示（即：用小段的<%%>），可读性很差，这样后期维护会比较麻烦，所以引入了Java标签库。

* **常用的JSTL标签库：**
  + **核心标签库：<http://java.sun.com/jsp/jstl/core>** 包含 Web 应用的常见工作，比如：循环、表达式赋值、基本输入输出等。
  + **格式化标签库：<http://java.sun.com/jsp/jstl/fmt>** 用来格式化显示数据的工作，比如：对不同区域的日期格式化等。
  + **函数标签库**：<http://java.sun.com/jsp/jstl/functions> 用来读取已经定义的某个函数。
  + **XML 标签库：<http://java.sun.com/jsp/jstl/xml>** 用来访问 XML 文件的工作，这是 JSTL 标签库的一个特点。 (用的很少)
  + **数据库标签库：**<http://java.sun.com/jsp/jstl/sql> **可以做访问数据库的工作。** (用的很少)

**J2EE 1.4:**

使用Java标签库首先要在项目中**引入jstl.jar和standard.jar两个jar包**，MyEclipse中有这个功能；



Standard.jar中有tld文件(是一个xml文档)，这是关于JSTL标签的说明文件，对应的java类就是加载这些说明，然后对我们写的JSTL标签进行解析处理。

**JAVAEE 5.0:**

**javaee5.0类库已经自带了jstl相关的jar包。不需要再添加，可以直接使用jstl。**

其次要在页面中加入这样一个标签头 **<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>**

<c:out>标签用于显示变量数据。

<c:out value="${a}" default="空" escapeXml="true" ></c:out>

<c:set>标签的常用属性（**将值存储到作用域或为 JavaBean 中的变量属性赋值** ）：



* <c:set var=“name“ value=“zhangsan“ scope=“page”/>  **（相当于：pageContext.setAttribute(“name”,”zhangsan”)）**
  + 将值zhangsan存储到变量name, name为作用域pageContext中的一个属性。
* <c:set var=“t” ><b>hello c:set!</b></c:set>
  + 将本体内容存储到变量t中。
* <c:set target="${u}" property="pwd" value="123" ></c:set>
  + 设置已有的EL变量u的属性pwd的值为123
* <c:set target="${u}" property="pwd" >123456</c:set>
  + 将本体内容设为EL变量u的属性pwd的值。

**<c:remove var="sampleValue" scope="session"/> 相当于：session.removeAttribute(“sampleValue”)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **c:if标签**🡪相当于java中的if语句   |  |  |  |  | | --- | --- | --- | --- | | 名称 | 描述 | 类型 | 默认值 | | test | 若返回true,则执行本体内容；否则，不执行。 | Boolean | 无。 | | var | 用来存储test执行后的结果 | String | 无 | | Scope | var 指定变量的存储范围 | String | page | |
| <c:if test="${empty requestScope.s}" var=”resultTest” >  <h1>true才显示</h1>  </c:if>  执行结果：${ resultTest } |

|  |
| --- |
| **c:choose标签**🡪相当于java中的if-else语句 |
| <c:choose>  <c:when test="${empty requestScope.s}">  <h1>空</h1>  </c:when>  <c:otherwise>  <h1>不空！</h1>  </c:otherwise>  </c:choose>  使用要点：   1. <c:when>必须在<c:otherwise>之前 2. <c:choose>中除了<c:when>/<c:otherwise>不能有其他元素。 |

|  |
| --- |
| **c:forEach标签**🡪相当于java中的遍历操作.  常用参数及其含义：    **varStatus属性的使用**： |
| <%  List list = new ArrayList();  for(int i=0;i<20;i++){  list.add("A:"+i);  }  pageContext.setAttribute("li",list);  %> |
| <c:forEach items="${li}" var="a" begin="5" end="13" step="3" varStatus="idx" >  <b>${a}</b>  <c:if test="${(idx.index+1)%5= =0}">  <br>  </c:if>  </c:forEach> |

|  |
| --- |
| <c:import url="http://www.baidu.com"></c:import>类似于 include。**但是比include范围广**。Include只能加载本web应用的路径。**Import可以加载本web应用以外的路径。**下面是几种典型的使用：  1.引入本web应用中的资源：  <c:import url="/1.jsp" ></c:import> 此处/代表本web应用根目录  2. 引入外部URL:  <c:import url="http://www.baidu.com" ></c:import>  3. 将所引入文件的内容作为字符串存储在变量中，留待以后使用。  <c:import url="1.jsp" var="t5" scope="page"></c:import>  ${t5} |
| 请求重定向：<% response.sendRedirect(“fortoken.jsp”) %>  <c:redirect url="fortoken.jsp">地址栏改变  <c:param name="a" value="33"></c:param>  <c:param name="b" value="44"></c:param>  </c:redirect> |
| ${param.a} |
| <%  String str = "Where you from?I am from China";  request.setAttribute("str",str);  %>  <c:forTokens items="${str}" delims="?| " var="str1">  <b>${str1}</b><br>  </c:forTokens> |

**Fn标签库的使用：**

**简介**

在jsp页面上经常遇到得到集合长度、字符长度、字符切取等应用需，在2.0以前这种需是许多程序员对JSTL及为不满意的地方之一。为此在2.0 中添加了functions标签，其提供对以上需求的支持。

**使用方法**

引用

<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn"%>

函数说明：

|  |  |
| --- | --- |
| **函数** | **描述** |
| fn:contains(string, substring) | 如果参数string中包含参数substring，返回true |
| fn:containsIgnoreCase(string, substring) | 如果参数string中包含参数substring（忽略大小写），返回true |
| fn:endsWith(string, suffix) | 如果参数 string 以参数suffix结尾，返回true |
| fn:escapeXml(string) | 将有特殊意义的XML (和HTML)转换为对应的XML character entity code，并返回 |
| fn:indexOf(string, substring) | 返回参数substring在参数string中第一次出现的位置 |
| fn:join(array, separator) | 将一个给定的数组array用给定的间隔符separator串在一起，组成一个新的字符串并返回。 |
| fn:length(item) | 返回参数item中包含元素的数量。参数Item类型是数组、collection或者String。如果是String类型,返回值是String中的字符数。 |
| fn:replace(string, before, after) | 返回一个String对象。用参数after字符串替换参数string中所有出现参数before字符串的地方，并返回替换后的结果 |
| fn:split(string, separator) | 返回一个数组，以参数separator 为分割符分割参数string，分割后的每一部分就是数组的一个元素 |
| fn:startsWith(string, prefix) | 如果参数string以参数prefix开头，返回true |
| fn:substring(string, begin, end) | 返回参数string部分字符串, 从参数begin开始到参数end位置，不包括end位置的字符 |
| fn:substringAfter(string, substring) | 返回参数substring在参数string中后面的那一部分字符串 |
| fn:substringBefore(string, substring) | 返回参数substring在参数string中前面的那一部分字符串 |
| fn:toLowerCase(string) | 将参数string所有的字符变为小写，并将其返回 |
| fn:toUpperCase(string) | 将参数string所有的字符变为大写，并将其返回 |
| fn:trim(string) | 去除参数string 首尾的空格，并将其返回 |

**扩展知识点：**

**JSTL的格式化标签库。（自学！！）**

<!-- jstl-fmt -->

02 <c:set var="date" value="<%=new Date() %>"/>

03 <fmt:formatDate value="${date }" pattern="yyyy年MM月dd日"/><!-- 2010年06月23日 -->

04 <fmt:formatDate value="${date }" type="date"/><!-- 2010-6-23 -->

05 <fmt:formatDate value="${date }" type="time"/><!-- 15:29:14 -->

06 <fmt:formatDate value="${date }" type="both"/><!-- 2010-06-23 15:29:14 -->

07 <fmt:formatDate value="${date }" type="date" dateStyle="default"/><!-- 2010-6-23 -->

08 <fmt:formatDate value="${date }" type="date" dateStyle="short"/><!-- 10-6-23 -->

09 <fmt:formatDate value="${date }" type="date" dateStyle="medium"/><!-- 2010-6-23 -->

10 <fmt:formatDate value="${date }" type="date" dateStyle="long"/><!-- 2010年6月23日 -->

11 <fmt:formatDate value="${date }" type="date" dateStyle="full"/><!-- 2010年6月23日 星期三 -->

12 <!-- num= --><br/>

13 <c:set var="num" value="1234567890.1234567"/>

14 <fmt:formatNumber value="${num }" type="number"/><!-- 1,234,567,890.123 -->

15 <fmt:formatNumber value="${num }" pattern="#" type="number"/><!-- 1234567890 -->

16 <fmt:formatNumber value="${num }" pattern="#.##" type="number"/><!-- 1234567890.12 -->

17 <fmt:formatNumber value="${num }" pattern="#.00" type="number"/><!-- 1234567890.12 -->

18 <fmt:formatNumber value="${num }" type="currency"/><!-- ￥1,234,567,890.12 -->

19 <fmt:formatNumber value="${num }" type="percent"/><!-- 123,456,789,012% --><br/>

20 <fmt:parseNumber var="i" type="number" value="-45678.2q345" />${i}<!-- -45678.2 忽略第一个不符合数字条件的字符和其之后的所有字符，如果字符串不是以数字开头则报错 -->

符号                  意义  
0                     一个数位  
#                     一个数位，前导零和追尾零不显示  
.                      小数点分割位置  
，                     组分隔符的位置  
-                      负数前缀  
%                    用100乘，并显示百分号  
其他任何符号    在输出字符串中包括指定符号

