**Java-JSTL（JSP标准标签库）介绍**

添加时间:2007-4-22

前言  
  
从JSP　1.1规范开始，JSP就支持在JSP中使用自定义标签了，自定义标签的广泛使用造成了程序员重复定义，这样就促成了JSTL（JavaServer Pages Standard Tag Library）的诞生。  
因为工作中需要用到JSTL，但网上却苦于找不到有关JSTL的中文资料，所以就有了这篇文章。  
  
JSTL简介  
  
JSTL是一个不断完善的开放源代码的JSP标签库，是由apache的jakarta小组来维护的。JSTL只能运行在支持JSP1.2和Servlet2.3规范的容器上，如tomcat 4.x。但是在即将推出的JSP 2.0中是作为标准支持的。  
JSTL目前的最新版本为1.02，最终发布版为1.0。JSTL包含两个部分：标签库和EL（Expression Language表达式语言）语言。标签库目前支持四种标签： 标签 URI 前缀 示例   
Core <http://java.sun.com/jstl/core> c <c:tagname ...>   
XML processing <http://java.sun.com/jstl/xml> x <x:tagname ...>   
I18N capable formatting <http://java.sun.com/jstl/fmt> fmt <fmt:tagname ...>   
Database access (SQL) <http://java.sun.com/jstl/sql> sql <sql:tagname ...>   
  
  
Core支持JSP中的一些基本的操作；  
XML processing支持XML文档的处理；  
I18N capable formatting支持对JSP页面的国际化；  
Database access (SQL)支持JSP对[数据库](http://www.ninedns.com/webhosting/MSSQL.ASP)的操作。  
  
由于本人水平有限，本文仅介绍Core标签，如有兴趣，可一起探讨其它三种标签的使用与扩充。  
  
EL语言介绍  
  
EL语言是JSTL输出（输入）一个JAVA表达式的表示形式。  
在JSTL中，EL语言只能在属性值中使用。EL语言只能通过建立表达式${exp1}来进行调用。在属性值中使用表达式有三种方式。  
  
1、 value属性包含一个表达式  
<some:tag value="${expr}"/>  
在这种情况下，表达式值被计算出来并根据类型转换规则赋值给value属性。比如：<c:out value="${username}" />中的${username}就是一个EL，它相当于JSP语句<%=request.getAttribute(“username”)%>或<%=session.getAttribute(“username”)%>  
  
2、 value属性包含一个或多个属性，这些属性被文本分割或围绕  
<some:tag value="some${expr}${expr}text${expr}"/>  
在这种情况下，表达式从左到右进行计算，并将结果转换为字符串型（根据类型转换规则），并将结果赋值给value属性  
  
3、 value属性仅仅包含文本  
<some:tag value="sometext"/>  
在这种情况下，字符串型属性value将根据类型转换规则转换为标签所希望的类型。  
  
EL语言的操作符  
取得某个对象或集合中的属性值  
为了获得集合中的属性，EL支持以下两种操作  
1． 使用.操作符来获得有名字的属性。例如表达式${user.username}表明对象user的username属性  
2． 使用[]操作符来获得有名字或按数字排列的属性。  
表达式${user["username"]}和表达式${user. username }含义相同  
表达式${row[0]} 表明row集合的第一个条目。  
在这里user是一个类的对象，它的属性username必须符合标准JavaBean的规范，即必须为username属性定义相应的getter、setter方法。  
  
Empty操作符(空值检查)  
  
使用empty操作符来决定对象、集合或字符串变量是否为空或null。例如：  
${empty param.username}  
如果request的参数列表中的username值为null，则表达式的值为true。 EL也可以直接使用比较操作符与null进行比较。如${param.firstname == null}。  
比较操作符 操作符 描述   
==或eq 相等检查   
!=或ne 不等检查   
<或lt 小于检查   
>或gt 大于检查   
<=或le 小于等于检查   
>=或ge 大于等于检查   
  
数字运算符与逻辑运算符均与JAVA语言相同，不再列表。  
  
Core标签库  
  
1、 通用标签  
  
<c:out>  
<c:out>标签用于在JSP中显示数据，它有如下属性 属 性 描 述 是否必须 缺省值   
value 输出的信息，可以是EL表达式或常量 是 无   
default  value为空时显示信息 否 无   
escapeXml 为true则避开特殊的xml字符集 否 true   
  
  
  
例子: 您的用户名是: <c:out value=”${user.username}” default=”guest”/>   
  
显示用户的用户名，如为空则显示guest  
<c:out value="${sessionScope.username}"/>   
  
指定从session中获取username的值显示；  
<c:out value="${username}" />   
  
显示username的值，默认是从request(page)中取，如果request中没有名为username的对象则从session中取，session中没有则从application(servletContext)中取，如果没有取到任何值则不显示。  
  
<c:set>  
<c:set>标签用于保存数据，它有如下属性 属 性 描 述 是否必须 缺省值   
value 要保存的信息，可以是EL表达式或常量 否    
target 需要修改属性的变量名，一般为javabean的实例 否 无   
property 需要修改的javabean属性 否 无   
var 需要保存信息的变量 否 无   
scope 保存信息的变量的范围 否 page   
  
如果指定了target属性, 那么property属性也必须指定。  
例子： <c:set value="${test.testinfo}" var="test2" scope=”session” />   
  
将test.testinfo的值保存到session的test2中，其中test是一个javabean的实例，testinfo是test对象的属性。  
<c:set target="${cust.address}" property="city" value="${city}"/>   
  
将对象cust.address的city属性值保存到变量city中  
  
<c:remove>  
<c:remove>标签用于删除数据，它有如下属性 属 性 描 述 是否必须 缺省值   
var 要删除的变量 是 无   
scope 被删除变量的范围 否 所有范围，包括page、request、session、application等   
  
例子: <c:remove var="test2" scope="session"/>   
  
从session中删除test2变量。  
  
2、 流控制标签  
  
<c:if>  
  
<c:if>标签有如下属性 属 性 描 述 是否必须 缺省值   
test 需要评价的条件，相当于if (...){}语句中的条件 是 无   
var 要求保存条件结果的变量名 否 无   
scope 保存条件结果的变量范围 否 page   
  
  
<c:choose>  
这个标签不接受任何属性  
  
<c:when>  
<c:when>标签有以下属性 属 性 描 述 是否必须 缺省值   
test 需要评价的条件 是 无   
  
  
<c:otherwise>  
这个标签同样不接受任何属性  
  
例子： <c:if test="${user.wealthy}">  
user.wealthy is true.  
</c:if>   
  
如果user.wealthy值true，则显示user.wealthy is true.  
  
<c:choose>  
<c:when test="${user.generous}">  
user.generous　is true.  
</c:when>   
<c:when test="${user.stingy}">  
user.stingy is true.  
</c:when>  
<c:otherwise>  
user.generous and user.stingy are false.  
</c:otherwise>  
</c:choose>   
  
只有当条件user.generous返回值是true时，才显示user.generous　is true.  
只有当条件user.stingy返回值是true时，才显示user.stingy　is true.  
其它所有的情况（即user.generous和user.stingy的值都不为true）全部显示user.generous and user.stingy are false.  
  
由于JSTL没有形如if (){…} else {…}的条件语句，所以这种形式的语句只能用<c:choose>、<c:when>和<c:otherwise>标签共同来完成了。  
  
3、 循环控制标签  
  
<c:forEach>  
<c:forEach>标签用于通用数据，它有以下属性 属 性 描 述 是否必须 缺省值   
items 进行循环的项目 否 无   
begin 开始条件 否 0   
end 结束条件 否 集合中的最后一个项目   
step 步长 否 1   
var 代表当前项目的变量名 否 无   
varStatus 显示循环状态的变量 否 无   
  
  
例子： <c:forEach items="${vectors}" var="vector">  
<c:out value="${vector}"/>  
</c:forEach>   
  
相当于java语句 for (int i=0;i<vectors.size();i++) {  
out.println(vectors.get(i));  
}   
  
在这里vectors是一个java.util.Vector对象，里面存放的是String数据，vector是当前循环条件下String对象。实际上这里的vectors可以是任何实现了java.util. Collection接口的对象。   
<c:forEach begin="0" end="100" var="i" step="1">  
count=<c:out value="${i}"/><br>  
</c:forEach>  
   
  
输出:  
count=0  
...  
count=100  
  
<c:forTokens>  
<c:forTokens>标签有以下属性 属 性 描 述 是否必须 缺省值   
items 进行循环的项目 是 无   
delims 分割符 是 无   
begin 开始条件 否 0   
end 结束条件 否 集合中的最后一个项目   
step 步长 否 1   
var 代表当前项目的变量名 否 无   
varStatus 显示循环状态的变量 否 无   
  
  
例子   
<c:forTokens items="a:b:c:d" delims=":" var="token">  
<c:out value="${token}"/>  
</c:forTokens>  
   
  
这个标签的使用相当于java.util.StringTokenizer类。在这里将字符串a:b:c:d以：分开循环四次，token是循环到当前分割到的字符串。  
  
4.导入文件和URL  
  
JSTL核心标签库支持使用<c:import>来包含文件，使用<c:url>来打印和格式化URL,使用<c:redirect>来重定向URL。  
  
<c:import>  
<c:import>标签包含另外一个页面代码到当前页，它有以下属性 属 性 描 述 是否必须 缺省值   
url 需要导入页面的url 是 无   
context /后跟本地web应用程序的名字 否 当前应用程序   
charEncoding 用于导入数据的字符集 否 ISO-8859-1   
var 接受导入文本的变量名 否 page   
scope 接受导入文本的变量的变量范围 否 1   
varReader 用于接受导入文本的java.io.Reader变量名 否 无   
varStatus 显示循环状态的变量 否 无   
  
  
<c:url>  
<c:url>标签输出一个url地址，它有以下属性 属 性 描 述 是否必须 缺省值   
url url地址 是 无   
context /后跟本地web应用程序的名字 否 当前应用程序   
charEncoding 用于导入数据的字符集 否 ISO-8859-1   
var 接受处理过的url变量名，该变量存储url 否 输出到页   
scope 存储url的变量名的变量范围 否 page   
  
  
例子：   
<c:import url="http://www.url.com/edit.js" var="newsfeed"/>  
   
  
将url http://www.url.com/edit.js包含到当前页的当前位置，并将url保存到newsfeed变量中   
<a href="<c:url url="/index.jsp"/>"/>  
   
  
在当前页的当前位置输出<a href="http://www.yourname.com/index.jsp"/>，http://www.yourname.com是当前页的所在的位置。  
  
  
<c:redirect>  
<c:redirect>标签将请求重新定向到另外一个页面，它有以下属性 属 性 描 述 是否必须 缺省值   
url url地址 是 无   
context /后跟本地web应用程序的名字 否 当前应用程序   
  
例子：   
<c:redirect url="http://www.yourname.com/login.jsp"/>  
   
  
将请求重新定向到http://www.yourname.com/login.jsp页，相当于response.setRedirect("http://www.yourname.com/login.jsp");  
  
<c:param>  
<c:param>标签用来传递参数给一个重定向或包含页面，它有以下属性 属 性 描 述 是否必须 缺省值   
name 在request参数中设置的变量名 是 无   
value 在request参数中设置的变量值 否 无   
  
例子：   
<c:redirect url="login.jsp">  
<c:param name="id" value="888"/>  
</c:redirect>  
   
  
将参数888以id为名字传递到login.jsp页面，相当于login.jsp?id=888  
  
  
JSTL的优点  
1、 在应用程序服务器之间提供了一致的接口，最大程序地提高了WEB应用在各应用服务器之间的移植。  
2、 简化了JSP和WEB应用程序的开发。  
3、 以一种统一的方式减少了JSP中的scriptlet代码数量，可以达到没有任何scriptlet代码的程序。在我们公司的项目中是不允许有任何的scriptlet代码出现在JSP中。  
4、 允许JSP设计工具与WEB应用程序开发的进一步集成。相信不久就会有支持JSTL的IDE开发工具出现。  
  
总结  
上面介绍的仅仅是JSTL的一部分，如果有时间我会继续把其它部分写出来分享给大家。如果要使用JSTL，则必须将jstl.jar和standard.jar文件放到classpath中，如果你还需要使用XML processing及Database access (SQL)标签，还要将相关JAR文件放到classpath中，这些JAR文件全部存在于下载回来的zip文件中。这个zip文件可以从http://jakarta.apache.org/builds/jakarta-taglibs/releases/standard/jakarta-taglibs-standard-1.0.zip下载。