JSP标签

1. **创建一个简单的自定义标签**
2. **第一步，创建一个标签处理器类：实现SimpleTag接口**

XxxSimpleTag，方法实现，先不写内容

1. **第二步，在WEB-IF文件夹下新建一个.tld（标签库描述文件）为拓展名的xml文件，并拷贝固定的部分，并对其信息修改**

<taglib xmlns=*"http://java.sun.com/xml/ns/j2ee"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-jsptaglibrary\_2\_0.xsd"*

version=*"2.0"*>

<!-- 描述信息 -->

<!-- 标签库版本信息 -->

<description>myTag 1.0 core library</description>

<!-- 标签库名 -->

<display-name>JSTL core</display-name>

<!-- 标签库版本信息 -->

<tlib-version>1.0</tlib-version>

<!-- 关键信息 -->

**<!-- 便签库建议使用前缀 -->**

<short-name>my</short-name>

**<!-- 标签库的uri是一个标签库的唯一标示 -->**

<uri>http://www.atguigu.com/mytag/core</uri>

</taglib>

1. **第三步在tld文件中描述自定义的标签**

<!-- 描述自定义标签信息 -->

<tag>

**<!-- 标签的名字 -->**

<name>hello</name>

**<!-- 标签所在的全类名 -->**

<tag-class>servlet.tag.HelloSimpelTag</tag-class>

**<!-- 标签体的类型 此为无标签体 -->**

<body-content>empty</body-content>

</tag>

1. **第四步在JSP页面上使用自定义标签**
   1. **在开头引入标签库**

<%@taglib uri=*"http://www.atguigu.com/mytag/core"*

prefix=*"my"*%>

* 1. **使用标签**

<my:hello/> //<前缀：标签名/>

1. **创建一个带属性的自定义标签**
2. **第一步，添加属性信息到tdl文件到对应tag节点中**

<attribute>

**<!-- 该属性的属性名 -->**

<name>value</name>

**<!-- 该属性是否为必须 -->**

<required>true</required>

**<!-- runtime expression value的合成词 ， 当前属性是否可以接受运行时表达式的动态值 -->**

<rtexprvalue>true</rtexprvalue>

</attribute>

1. **第二步，完善标签处理器类**
2. **添加一个PageContext属性**

**private** PageContext pageContext = **null**;

在setJspContext(JspContext pageContext)方法中设置这个属性

**this**.pageContext = (PageContext) pageContext;

JspContext是PageContext的父类，当服务器调用这个标签时，会把jsp页面的pageContext(JSP的上下文对象)传进来，就可以获取JSP页面的一切信息，可以获取其他8个jsp的隐含对象。在调用doTag()方法之前使用

1. **在类中添加标签中属性相对应的属性**

**private** String value;

**要为属性设置set方法**，JSP引擎会自动通过方法为其赋值，**建议把所有属性设置为String类型**

**public** **void** setValue(String value) {

**this**.value = value;

}

1. **完善doTag()方法**

标签具体执行操作的方法

**public** **void** doTag() **throws** JspException, IOException {

pageContext.getOut().print(value);

}

1. **第三步，带属性自定义标签使用**

<my:hello value=*"好！"*/>

1. **带标签体的自定义标签**
2. **第一步， 把标签添加到配置文件**

<tag>

<name>fargment</name>

<tag-class>servlet.tag.TestJspFragment</tag-class>

**<!-- 标签体的类型，设置为含有标签体 -->**

<body-content>scriptless</body-content>

</tag>

1. **第二步，完善标签处理类**

**类直接继承于SimpleTagSupport类，比较方便**

**完善doTag()方法**

1. **获取标签体的对象**

JspFragment fragment = getJspBody();

JspFragment，用于操作标签体内容的类，在jsp引擎调用该标签时，若配置了标签含有标签体，则jsp引擎会调用setJspBody()方法，把JspFragment传递给标签处理类。通过继承SimpleTagSupport类，已经写好的getJspBody()方法，可以直接获取JspFragment对象。

1. **创建一个StringWrite的对象**

StringWriter sw = new StringWriter();

该对象是一个字符流，可以用其回收在字符串缓冲区中的输出来构造字符串。可以存放输出流输出的字符串。

1. **通过invoke()方法获取方法体的内容**

fragment.invoke(sw);

invoke(Write) 传入一个Write对象，标签体内容输出的位置。如果传入null则，则输出到getJspContext().getOut()这个位置 即 输出到浏览器上。

**例**

实现 forEach 标签:

<c:forEach items="${requestScope.customers }" var="cust2">

${pageScope.cust2.id } -- ${cust2.name } <br>

</c:forEach>

**分析：**

> 两个属性: items(集合类型, Collection), var(String 类型)

> doTag:

\* 遍历 items 对应的集合

\* 把正在遍历的对象放入到 pageContext 中, 键: var, 值: 正在遍历的对象.

\* 把标签体的内容直接输出到页面上.

1. **带父标签的自定义标签**
   1. 父标签感受不到子标签的存在（无法获取子标签），仅把子标签当做标签体来使用，所以父标签的类型要设置为scriptless。
   2. 子标签可以通过getParent()获取父标签的引用（需要实现Simpletag接口或者继承SimopleTagSupport类）
   3. 若一个标签有父标签，则jsp引擎在调用标签时会把父标签的引用传给标签处理器，通过setParent赋值。（如果没有父标签parent为初值null）
2. **获取父标签**

JspTag jspTag=getParent();

1. **强转为父标签原本的类型**

设父标签类型为ParentTag

ParentTag parentTag=(Parent)jspTag;

1. **获取父标签信息进行操作**

略

**例**

实现

<c:choose>

<c:when test="${param.age > 24}">大学毕业</c:when>

<c:when test="${param.age > 20}">高中毕业</c:when>

<c:otherwise>高中以下...</c:otherwise>

</c:choose>

**分析：**

> 开发 3 个标签: choose, when, otherwise

> 其中 when 标签有一个 boolean 类型的属性: test

> choose 是 when 和 otherwise 的父标签

> when 在 otherwise 之前使用

> **在父标签 choose 中定义一个 "全局" 的 boolean 类型的 flag**: 用于判断子标签在满足条件的情况下是否执行.

**三种情况：**

\* 若 when 的 test 为 true, 且 when 的父标签的 flag 也为 true, 则执行 when 的标签体(正常输出标签体的内容), 同时把 flag 设置为 false

\* 若 when 的 test 为 true, 且 when 的父标签的 flag 为 false, 则不执行标签体.

\* 若 flag 为 true, otherwise 执行标签体.

1. **补充：**
   1. **继承SimpleTagSupport类**

该类类似与HttpServlet与Servlet。把SimpleTag接口中方法全部实现，但是不填写内容，需要的方法重写即可。JspContext对象被这个类获取，并提供getJspContext()方法来获取，使用时强转为PageContext类型即可使用。

* 1. **标签的类型有三种**
     + 1. **empty：**

没有标签体

* + - 1. **★scriptless：**

标签体可以包含el表达式和jsp动作元素，但不能包含jsp脚本元素（<%= %>就是一个脚本元素，带大于号小于号的一般都是脚本元素），最常用。

* + - 1. **tagdependent：**

表示标签体交由标签本身去解析处理。若指定tagdependent，在标签体中的所有代码都会原封不动的交给标签处理器处理，而不是将执行结果交给标签处理器。（不会执行便签体里面的内容，**直接将里面的字符串提交给标签处理器**）

* 1. **调用标签需要的包**

**standard.jar jstl.jar**

**常用标签的uri：**[*http://java.sun.com/jsp/jstl/core*](http://java.sun.com/jsp/jstl/core)

**（foreach、choose等）**