[**[教程]自定义JSP中的Taglib标签之三简单仿JSTL中带标签体的ForEach循环**](http://www.cnblogs.com/edwardlauxh/archive/2010/05/19/1918590.html)

上2篇文章分别了介绍了taglib的无属性状态和有带属性状态,但是都是分别都是独立的闭标签,这次带来的是带属性的和带标签体的例子,仿照jstl中的forEach循环构造一个简单的循环标签.

Java代码如下:

package org.lxh.taglib;

import java.util.Collection;

import java.util.Iterator;

import java.util.Map;

import javax.servlet.jsp.JspException;

import javax.servlet.jsp.tagext.BodyTagSupport;

public class TestForEach extends BodyTagSupport {

private static final long serialVersionUID = -5210136338413610356L;

private String var;

private Iterator<?> iterator;

public String getVar() {

return var;

}

public void setVar(String var) {

this.var = var;

}

public void setItem(Object item) {

if (item instanceof Map) {

Map items = (Map) item;

this.iterator = items.entrySet().iterator();

} else {

Collection<?> c = (Collection) item;

this.iterator = c.iterator();

}

}

@Override

public int doAfterBody() throws JspException {

if (this.process()) {

return EVAL\_BODY\_AGAIN;

} else {

return EVAL\_PAGE;

}

}

@Override

public int doStartTag() throws JspException {

if (this.process())

return EVAL\_BODY\_INCLUDE;

else

return EVAL\_PAGE;

}

private boolean process() {

if (null != iterator && iterator.hasNext()) {

Object item = iterator.next();

pageContext.setAttribute(var, item);

return true;

} else

return false;

}

}

问题1:为什么要继承BodyTagSupport 而不去只实现IterationTag接口或者直接继承TagSupport使用呢?

TagSupport与BodyTagSupport的区别主要是标签处理类是否需要与标签体交互，如果不需要交互的就用TagSupport，否则如果不需要交互就用BodyTagSupport。  
交互就是标签处理类是否要读取标签体的内容和改变标签体返回的内容。用TagSupport实现的标签，都可以用BodyTagSupport来实现，因为BodyTagSupport继承了TagSupport而不去实现IterationTag接口的,因为BodyTagSupport继承了TagSupport类,并且该类已经实现了IterationTag接口并且实现了功能.

doStartTag()方法在标签开始时执行，要记住每次都要对类进行初始化，避免上一次的遗留数据对操作造成影响。然后判断是否有数据需要处理，如果有，则返回EVAL\_BODY\_INCLUDE开始处理标签里的内容，如果没有，返回 EVAL\_PAGE跳过标签内容执行标签下面的内容。

doAfterBody()方法在每次处理完标签内部内容后执行，判断循环是否已经结束，如果可以继续循环，返回EVAL\_BODY\_AGAIN用循环得到新的数据再次处理标签内部内容，如果循环结束就返回EVAL\_PAGE结束标签。

tld如下:

<?xml version="1.0" encoding="UTF-8"?>

<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">

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

<short-name>for</short-name>

<uri></uri>

<tag>

<name>foreach</name>

<tag-class>org.lxh.taglib.TestForEach</tag-class>

<tei-class>org.lxh.tagei.ForEachInfo</tei-class>

<body-content>JSP</body-content>

<attribute>

<name>var</name>

<required>true</required>

<rtexprvalue>true</rtexprvalue>

</attribute>

<attribute>

<name>item</name>

<rtexprvalue>true</rtexprvalue>

<type>java.lang.Object</type>

<!-- <type>java.util.Collection</type>-->

</attribute>

</tag>

</taglib>

是不是发现tld有点改变了,没错,如果要使用包含标签体的标签库的话需要在<body-content>标签中加入jsp,这样的话该标签表示可以包含jsp和html的标签进行一起循环,上面是否还发现多了一个<tei-class>标签,该标签用途就是对当前自定义的标签体进行验证检查的.

tei验证检查代码如下:

package org.lxh.tagei;

import javax.servlet.jsp.tagext.TagData;

import javax.servlet.jsp.tagext.TagExtraInfo;

import javax.servlet.jsp.tagext.VariableInfo;

public class ForEachInfo extends TagExtraInfo {

final private static String ITEMS = "item";

final private static String VAR = "var";

@Override

public VariableInfo[] getVariableInfo(TagData data) {

return new VariableInfo[] {

new VariableInfo(data.getAttributeString("var"),

"java.lang.String", true, VariableInfo.AT\_BEGIN),

new VariableInfo(data.getAttribute("item").toString(),

"java.lang.Object", true, VariableInfo.AT\_BEGIN) };

}

@Override

public boolean isValid(TagData us) {

if (!Util.isSpecified(us, ITEMS))

return false;

return true;

}

}

public class Util {

/\*\*

\* Returns true if the given attribute name is specified, false otherwise.

\*/

public static boolean isSpecified(TagData data, String attributeName) {

return (data.getAttribute(attributeName) != null);

}

才用验证的好处非常之多的,可以对当前传入的值进行验证,还有非空的检查,在标签的时候如果不写某个标签体会出现验证为false的提示.  
jsp代码:

<%@ page language="java" contentType="text/html; charset=UTF-8"  
    pageEncoding="UTF-8"%>  
<%@taglib prefix="c" uri="/WEB-INF/tld/testforeach.tld"%>  
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "[http://www.w3.org/TR/html4/loose.dtd"](http://www.w3.org/TR/html4/loose.dtd%22)>

<%@page import="java.util.\*"%>

<html>  
<head>  
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
<title>Insert title here</title>  
</head>  
<%  
    List<String> list = new ArrayList<String>();  
    list.add("aa");  
    list.add("bb");  
    list.add("cc");  
    Map map = new HashMap();  
    map.put("1","a");  
    map.put("2","b");  
    map.put("3","c");  
    map.put("4","b");  
%>  
<body>  
<table>

    <c:foreach var="hi"  item="<%=map %>" >  
        <tr>  
            <td>${hi}</td>  
        </tr>  
    </c:foreach>  
</table>  
</body>  
</html>