**Struts2标签**

·页面标签概述

表单标签将在HTML文档里面呈现为一格表单元素

使用表单标签的优点：表单回显。

标签的属性可以被赋值为一个静态的值或一个OGNL表达式。**如果在赋值时使用了一个OGNL表达式并把它用%{}括起来，这个表达式将会被求值。**

参考：docs/WW/docs/tag-reference.html

·property类标签

范例：**s:property标签测试**

**package** wp.biaoqiandemo;

**import** com.opensymphony.xwork2.ActionContext;

**import** com.opensymphony.xwork2.ActionSupport;

**public** **class** TagAction **extends** ActionSupport{

@Override

**public** String execute() **throws** Exception {

//往值栈中放入数据

ActionContext.*getContext*().put("name", "文攀");

**ActionContext.*getContext*().put("table", "<table border = '1'><tr><td>HTML标签</td></tr></table>") ;**

**return** ***SUCCESS***;

}

}

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

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns=*"http://www.w3.org/1999/xhtml"*>

<head>

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

<%@ taglib prefix = *"s"* uri=*"/struts-tags"* %>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"* />

<title>s:property标签测试</title>

</head>

<body>

<h3>利用s:property 从map栈中获取数据</h3>

<s:property value = *"#name"*/>

<h3>s:property 的default属性(若没有此属性则取默认值)</h3>

<s:property value=*"#age"* default=*"0"*/>

<h3>s:property escapeHtml属性，是否对HTML标签进行转义，默认不转义（true）</h3>

**<s:property escapeHtml=*"false"* value = *"#table"*/>**

</body>

</html>

在以上程序中可以看出：s:property标签用于输出value的内容。

Default属性为，当没有要输出的属性时就输出默认的值。

escapeHtml属性的作用是将HTML代码进行转义。

**·iterator数据类标签（非常重要）**

作用：遍历集合对象（可以是list可以是set和数组等），显示集合对象的数据（跟<c:foreach>功能一样）

属性：

Value:迭代集合，支持OGNL表达式，如果没有设置该属性，则默认使用值栈栈顶的集合来迭代。

Var:引用变量的名称，该变量是集合迭代时的子元素。

Struts：引用迭代时的状态对象IteratorStatus实例，其有如下几个方法：

·int getCount():返回当前迭代了几个元素；

·int getIndex():返回当前迭代元素的索引

·Boolean odd:返回当前迭代元素的索引是否是奇数

Boolean even：返回当前迭代的元素的索引是否是偶数

Boolean isFirst():返回当前迭代的元素的索引是否是第一个

Boolean isLast():返回当前迭代元素的索引是否是最后一个

范例：在jsp页面使用循环方式输出1---10（打印1--10）

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

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns=*"http://www.w3.org/1999/xhtml"*>

<head>

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

<%@ taglib prefix = *"s"* uri=*"/struts-tags"* %>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"* />

<title>s:iterator标签测试</title>

</head>

<body>

<h3>使用s:iterator打印1-10数字</h3>

<h4>把遍历每一个值存在map栈中，变量名定义为num</h4>

<h4>把每个遍历到的值也存在root栈中</h4>

<h4>把每一个遍历到的值的状态也存在map栈中，定义变量名为s</h4>

<s:iterator begin=*"1"* end=*"10"* var = *"num"* status=*"s"*>

<s:property value = *"#num"*/>--

<s:property value = *"top"*/>--

<s:property value = *"[0].top"*/>--

<s:property value = *"#s.index"*/>--

<s:property value = *"#s.count"*/>--

<s:property value = *"#s.odd"*/>--

<s:property value = *"#s.even"*/>--

<s:property value = *"#s.isFirst()"*/>--

<s:property value = *"#s.isLast()"*/>--

<br></br>

</s:iterator>

</body>

</html>

在以上程序中，利用s:iterator标签进行了1--10的数字遍历，将每个遍历到的数字都存放在num变量中，然后通过<s:property>标签进行输出

范例：利用s:iterator标签遍历集合

**package** wp.tagdemo;

**import** java.util.ArrayList;

**import** java.util.List;

**import** com.opensymphony.xwork2.ActionContext;

**import** com.opensymphony.xwork2.ActionSupport;

**public** **class** TagAction1 **extends** ActionSupport{

@Override

**public** String execute() **throws** Exception {

List<User> users = **new** ArrayList<>() ;

users.add(**new** User("tom","123")) ;

users.add(**new** User("Ami","321")) ;

users.add(**new** User("chec","456")) ;

//把数据放入值栈

ActionContext.*getContext*().getValueStack().set("users", users);

**return** ***SUCCESS***;

}

}

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

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns=*"http://www.w3.org/1999/xhtml"*>

<head>

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

<%@ taglib prefix = *"s"* uri=*"/struts-tags"* %>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"* />

<title>s:iterator标签测试</title>

</head>

<body>

<h3>s:iterator遍历集合对象</h3>

<h4>s:iterator value属性就是用来获取值栈中的集合对象</h4>

<s:iterator value=*"users"* var = *"u"*>

<h4>使用map栈取值</h4>

<s:property value = *"#u.username"*/>--->

<s:property value = *"#u.pwd"*/><br></br>

<h4>使用root栈取值</h4>

<s:property value = *"top.username"*/>--->

<s:property value = *"top.pwd"*/><br></br>

<s:property value = *"[0].username"*/>--->

<s:property value = *"[0].pwd"*/><br></br>

</s:iterator>

</body>

</html>

在以上程序中首先利用action将user对象加入list集合，然后在jsp页面中利用s:iterator标签将集合中的值逐个遍历。

**·if数据类标签**

<s:if> <s:elseif> <s:else>标签：**作用是页面判断，其中的test属性可以接收OGNL表达式**

范例：根据后台传入的用户状态值(0.1.2)，在页面判断并显示中文（管理员，普通用户，游客）

package wp.tagdemo;

import java.util.ArrayList;

import java.util.List;

import com.opensymphony.xwork2.ActionContext;

import com.opensymphony.xwork2.ActionSupport;

public class TagAction1 extends ActionSupport{

//利用属性参数接收

private String role ;

public String getRole() {

return role;

}

public void setRole(String role) {

this.role = role;

}

@Override

public String execute() throws Exception {

//把role值放入值栈

ActionContext.getContext().getValueStack().set("role",role );

return SUCCESS;

}

}

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

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns=*"http://www.w3.org/1999/xhtml"*>

<head>

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

<%@ taglib prefix = *"s"* uri=*"/struts-tags"* %>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"* />

<title>s:if标签测试</title>

</head>

<body>

<h3>s:if标签</h3>

<s:if test=*"role==0"*>

<h3>管理员</h3>

</s:if>

<s:elseif test=*"role==1"*>

<h3>普通用户</h3>

</s:elseif>

<s:else>

<h3>游客</h3>

</s:else>

</body>

</html>

·a超链接类标签

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

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns=*"http://www.w3.org/1999/xhtml"*>

<head>

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

<%@ taglib prefix = *"s"* uri=*"/struts-tags"* %>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"* />

<title>s:a标签测试</title>

</head>

<body>

<h3>s:a超链接标签</h3>

<s:a action = *"tag3"* namespace = *"/"*>

<s:param name = *"role"*>0</s:param>

tag3.action</s:a>

</body>

</html>

在以上程序中：

**<s:a action = "tag3" namespace = "/">**

**<s:param name = "role">0</s:param>**

**tag3.action</s:a>**

Action表示点击后要跳转的action，**<s:param name = "role">0</s:param>** 为属性标签，是为了跳转的action中的属性赋一个值。

·用户界面（UI）标签

范例：

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

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns=*"http://www.w3.org/1999/xhtml"*>

<head>

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

<%@ taglib prefix = *"s"* uri=*"/struts-tags"* %>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"* />

<title>s:a标签测试</title>

</head>

<body>

<h3>s:form标签</h3>

<s:form action = *"tag3"* method=*"post"*>

<h3>文本框<s:textfield name = *"username"*></s:textfield></h3>

<h3>密码框<s:password name = *"pwd"*></s:password></h3>

<h3>隐藏域<s:hidden name = *"hide"*></s:hidden></h3>

<h3>文本框<s:textarea name = *"area"*></s:textarea></h3>

<hr />

<h3>单选按钮1<s:radio list=*"{'男','女'}"* name = *"sex"*></s:radio></h3>

<h3>value与值不一样</h3>

<h3>单选按钮2<s:radio list=*"*#{'m':'男','f':'女'}*"* name = *"sex"*></s:radio></h3>

<h3>复选框</h3>

<s:checkboxlist list=*"*#{'aa':'篮球','bb':'足球','cc':'羽毛球'}*"* name=*"check"*></s:checkboxlist>

<h3>下拉选择框

<s:select list=*"*#{'a':'上海','b':'成都','c':'北京' }*"*></s:select>

</h3>

<s:submit value = *"提交"*></s:submit>

<s:reset value = *"重置"*></s:reset>

</s:form>

</body>

</html>

在以上的用户界面标签测试中，复选框和单选框必须要写name属性，不然会报错。若要使下拉选择框单选框复选框的value值和显示的值不一样，则list中使用#{‘value值’:’显示的值’}的形式。

**·主题样式**

在以上的用户界面设计中可以发现文字与文本框等组件是完全分离的，因为在struts中默认使用的是xml主题。需要在struts.xml配置文件中重新配置主题常量。

<!-- 修改主题风格为简单主题 -->

<constant name = *"struts.ui.theme"* value = *"simple"*></constant>

表示使用的是simple（简单）主题。而不是xml主题。