Struts2标签库简介

# 概述

Struts2框架的标签库主要分为三类：

* 用户界面标签（UI标签）
  + 表单标签
  + 非表单标签
* 非用户界面标签（非UI标签）
  + 数据访问标签
  + 流程控制标签
* AJAX标签：Asynchronous Javascript And XML(异步JavaScript和XML)
* 要在JSP页面中引用Struts2标签库，需要使用taglib指令。在JSP代码的顶部加入以下的代码：

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

# Struts标签

## Struts2标签的作用

* 尽量避免在JSP页面中使用Java代码，而改用类似HTML的标签的形式来表达页面逻辑，让逻辑与显示分离，提高JSP的可维护性；
* 由于HTML自身的标签的表达能力不足，通过使用JSP Tag，可以对HTML语义进行扩展，从而完成解决页面显示数据、封装简单页面逻辑等HTML自身标签无法完成的工作；

## Struts2标签简介

* Struts 2框架使用OGNL作为默认的表达式语言
* OGNL是Object Graphic Navigation Language（对象图导航语言）的缩写，是一个开源项目
* ValueStack(值栈)是Struts 2中OGNL的根对象
* 访问其他Context中的对象，由于不是根对象，在访问时，需要加#前缀

## <s:property>标签

* **语法：<s:property value=*"msg"*/>**
* **用途：**可以获取Action类中msg属性的值
  + value属性实际上指向了Action类中的成员msg，而非”msg”这个字符串。例如，假定Action类中的数据成员msg为String类型，当msg=“Hello!”时，页面上将显示” Hello!” 字符串而非”msg”。
* **原理：**实际上，**<s:property>**标签是通过Action类的getMsg()方法来获取msg的值的
  + 如果Action类中没有定义该方法，则无法获取msg的值；
  + 在Action类中的getMsg()方法不能写作：getmsg()方法

## 数据的三种传递方式

* JSP🡪Action类
* 通过Action直接访问JSP页面，使用<s:property>标签访问Action类中成员变量的值
* JSP页面🡪Action🡪JSP页面

### 数据的传递方式1：JSP🡪Action类

### 数据的传递方式2：Action🡪JSP，并使用<s:property>标签访问Action类中成员变量的值

**OgnlAction.java**

**package com;**

**public class OgnlAction extends ActionSupport**{

**private String msg;**

**public String getmsg() {**

System.*out.println("getmsg() is called!");*

**return msg;**

}

**public String getMsg() {**

System.*out.println("getMsg() is called!");*

**return msg;**

}

}

**ognl.jsp**

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

**<p>The value of "msg" is <s:property value=*"msg"*/></p>**

**struts.xml**

**<action name=*"ognl" class="com.OgnlAction">***

**<result>/ognl.jsp</result>**

**</action>**

* 如果要获取msg的值，ognl.jsp页面必须与Action类关联
* 测试：
  + 通过action访问
  + 直接访问ognl.jsp页面

### 数据的传递方式3：JSP🡪Action🡪JSP

**struts.xml**

<action name=*"login"* class=*"com.Login"*>

<result name=*"success"*>**/welcome.jsp**</result>

<result name=*"failed"*>/error.jsp</result>

</action>

**index.jsp**

<**s:form** theme=*"simple"* method=*"post"* namespace=*"/Test"* action=*"login!login"*>

<p>用户名：<**s:textfield** name=*"username"* label=*"用户名"*/></p>

<p>密 码：<**s:password** name=*"password"* label=*"密码"* maxlength=*"30"*/></p>

<p><**s:submit** value=*"提交"*/></p>

<**/s:form**>

**Login.java**

**public** String login(){

System.*out*.println("login() is called!");

System.*out*.println(password);

**if**(username.equals("zjut"))

**return** "success";

**return** "failed";

}

**welcome.jsp**

**<p>欢迎您， <s:property value=*"username"* />！ </p>**

## Struts2用户登录模块处理流程概要

1. **浏览器发送请求http://localhost/Appname/login.action，到web应用服务器；**
2. **容器接收到该请求，根据web.xml的配置，服务器将请求转到FilterDispatcher进行处理，进入struts2的流程中；**
3. **框架在struts.xml配置文件中查找名为login的action对应的类；**
4. **框架初始化该Action，执行其中的set\*\*\*方法填充数据，再执行execute方法；**
5. **execute方法执行后并返回字符串“success”或者“fail”，分别表示登录成功或者登录失败；**
6. **框架检查配置以查看当返回的字符串所对应的结果页面**

## <s:checkbox>、<s:checkboxlist>与Action类中的String[]数据成员

### <s:checkbox>：<http://localhost:8080/chp3/checkbox.jsp>

<s:form action=*"checkbox.action" theme="simple">*

<s:checkbox name=*"hobbies"* fieldValue=*"篮球"*>篮球</s:checkbox>

<s:checkbox name=*"hobbies"* fieldValue=*"绘画"*>绘画</s:checkbox>

<s:checkbox name=*"hobbies"* fieldValue=*"旅游"*>旅游</s:checkbox>

<s:checkbox name=*"hobbies"* fieldValue=*"书法"* value=*"true"*>书法</s:checkbox>

<s:submit value=*"提交"/>*

</s:form>

### <s:checkboxlist>

1. **直接用Collection或Map批量设置数据：**[**http://localhost:8080/chp3/checkboxlist.jsp**](http://localhost:8080/chp3/checkboxlist.jsp)

* <s:checkboxlist name=*"a" label="请选择您喜欢的图书" labelposition="top"* list=*"{'Struts2权威指南' , '轻量级J2EE企业应用实战' , '基于J2EEAjax宝典'}" />*
* <s:checkboxlist name=*"b" label="请选择您想选择出版日期" labelposition="top"*

list="#{'Struts2权威指南':'2007年10月','轻量级J2EE企业应用实战':'2007月4月','基于J2EE的Ajax宝典':'2007年6月'}" listKey=*"key" listValue="value" />*

1. **从Action中获取数据：**[**http://localhost:8080/chp3/checkbox1**](http://localhost:8080/chp3/checkbox1)

* <s:checkboxlist name=*"book" label="请选择您喜欢的图书" labelposition="top"*

**list=*"#request.books"*** */>****// OGNL(Object Graph Navigation Language)***

* <s:checkboxlist name=*"book" label="请选择您喜欢的图书2" labelposition="top"*

**list=*"books"*** */>*

## <s:iterator>标签

在上下文中产生IteratorStatus的实例。

要遍历的可枚举的数据源

* **遍历页面中的集合**

**<table border=*"1"* width=*"200"*>**

**<s:iterator value=*"{'Struts2','Hibernate','Spring'}"* id=*"namex"* status=*"st"*>**

**<tr>**

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

**<td><s:property ~~value=~~*~~"namex"~~*/></td>**

**</tr>**

**</s:iterator>**

**</table>**

* **遍历action中的集合属性的值：**[**http://localhost:8080/chp3/showclass.action**](http://localhost:8080/chp3/showclass.action)

## <s:append>标签：<http://localhost:8080/chp3/append.jsp>

**<s:append id=*"newList"*>**

**<s:param**

**value="#{'Spring2.0宝典':'Tom','轻量级J2EE企业应用实战':'Jack','算法':'李刚'}" />**

**<s:param value="#{'新东方IT培训', '东方标准职业教育'}" />**

**</s:append>**

**<table border=*"1"* width=*"240"*>**

**<s:iterator value=*"#newList"* status=*"st"*>**

**<tr <s:if test=*"#st.odd"*>style="background-color:#bbbbbb"</s:if>>**

**<td><s:property value=*"key"*/></td>**

**<td><s:property value=*"value"*/></td>**

**</tr>**

**</s:iterator>**

**</table>**

## <generator>标签

**<s:generator val=*"'abcdef,Struts,Hibernate,Spring'"* separator=*","*>**

**<s:iterator id=*"namex"* status=*"st"*>**

**<tr>**

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

**<td><s:property value=*"namex"*/></td>**

**</tr>**

**</s:iterator>**

**</s:generator>**

## <sort>标签：<http://localhost:8080/chp3/sort.jsp>

**<s:bean id=*"mycomparator"* name=*"com.MyComparator"* />**

**<table border=*"1"* width=*"200"*>**

**<!-- source属性对应的是用于排序的字符串 -->**

**<s:sort source=*"{'Spring2.0', 'J2EE', 'abcdefghij', 'struts2'}"***

**comparator=*"#mycomparator"*>**

**<s:iterator status=*"st"*>**

**<tr <s:if test=*"#st.odd"*>style="background-color:#bbbbbb"</s:if>>**

**<td><s:property /></td>**

**</tr>**

**</s:iterator>**

**</s:sort>**

**</table>**

|  |
| --- |
| **package com;**  **import java.util.Comparator;**  **public class MyComparator implements Comparator {**  **public int compare(Object element1, Object element2) {**  **return element1.toString().length() - element2.toString().length();**  **}**  **}** |

## <set>标签：<http://localhost:8080/chp3/set.action>

**<s:set name=*"usr1"* value=*"#request.user"* />**

**<s:set name=*"usr"* value=*"#request['user']"* />**

**<!--获取user的name属性 -->**

**User对象的name值为：**

**<s:property value=*"#request.user.name"*/><br>**

**<s:property value=*"user.name"*/><br>**

**<s:property value=*"#usr1.name"* /><br>**

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

**<!--获取user的password属性 -->**

**User对象的password值为：<s:property value=*"#usr.password"*/><br>**

## <bean>标签：<http://localhost:8080/chp3/bean.jsp>

**<h2>bean示例</h2>**

**<!-- 使用bean标签 -->**

**<s:bean name=*"com.User"* id=*"usr"*>**

**<!-- 使用param标签直接给bean实例设置值 -->**

**<s:param name=*"name"* value=*"'admin'"*></s:param>**

**<s:param name=*"password"* value=*"123"*></s:param>**

**<!-- 使用property标签直接输出bean属性值 -->**

**属性name的值：<s:property value=*"name"*/><br/>**

**属性password的值：<s:property value=*"password"*/><br/>**

**</s:bean>**

**<h2>在bean标签的外部访问bean的属性</h2>**

**属性name的值：<s:property value=*"#usr.name"*/><br/>**

**属性password的值：<s:property value=*"#usr.password"*/><br/>**

## <include>标签：<http://localhost:8080/chp3/include.jsp>

***include.jsp***

**<s:include value=*"include\_file.jsp"*>**

**<s:param name=*"author"* value=*"'这是被包含的页面'"* />**

**<s:set name=*"msg"* value=*"'Hi, how are you!'"* />**

**</s:include>**

***include\_file.jsp***

**<body>**

**Hi, 您好!**

**<!--表达式语言输出author的请求参数-->**

**${param.author}**

**${msg}**

**</body>**

## <date>标签：<http://localhost:8080/chp3/date.jsp>

**<% //生成一个Date实例**

**java.util.Date now = new java.util.Date(107, 12, 23, 13, 23, 24);**

**//将该Date实例设置成一个pageContext里的属性**

**pageContext.setAttribute("now", now);**

**%>**

**<h5>nice = "false"，且指定format=“dd/MM/yyyy”</h5>**

**<s:date name=*"#attr.now"* format=*"dd/MM/yyyy"* nice=*"false"* /><br/>**

**<s:date name=*"#attr.now"* format=*"yyyy/MM/dd"* nice=*"false"* />**

**<hr>**

**<h5>nice = "true"，且指定format=“dd/MM/yyyy”</h5>**

**<s:date name=*"#attr.now"* format=*"dd/MM/yyyy"* nice=*"true"* />**

**<hr>**

**<h5>指定nice = "true"</h5>**

**<s:date name=*"#attr.now"* nice=*"true"* />**

**<hr>**

**<h5>nice = "false"，且没有指定format</h5>**

**<s:date name=*"#attr.now"* nice=*"false"* />**