1. **Ognl表达式基本原理和使用方法**
2. **Ognl表达式基本原理和使用方法**
3. **1.Ognl表达式语言**
4. **1.1.概述**

**OGNL表达式**  
OGNL是Object Graphic Navigation Language(对象图导航语言)的缩写，他是一个开源项目。Struts框架使用OGNL作为默认的表达式语言。  
**OGNL优势**

* 支持对象方法调用，如：×××.doSomeSpecial();
* 支持类静态的方法调用和值访问，表达式的格式

@[类全名（包括包路径）]@[方法名 |  值名]，例如：  
@java.lang.String@format('foo %s', 'bar')  
或@tutorial.MyConstant@APP\_NAME；

* 支持赋值操作和表达式串联，

如price=100, discount=0.8,calculatePrice()，这个表达式会返回80；

* 访问OGNL上下文（OGNL context）和ActionContext；
* 操作（创建）集合对象。

**总结：**OGNL 有一个上下文（Context）概念，说白了上下文就是一个MAP结构，它实现了java.utils.Map 的接口。

Struts框架默认就支持Ognl表达式语言。(从struts项目必须引入ognl.jar包可以看出)

**Ognl表达式语言的作用：**

* jsp页面取值用
* EL表达式语言，也用于页面取值，是jsp页面取值的标准（默认就可以使用）
* Ognl表达式语言，Struts标签默认支持的表达式语言，必须配置Struts标签用，不能离开Struts标签直接使用，就是说Ognl必须在Struts中使用
* 对比来看，EL使用范围更广，项目中不限制使用哪一种，哪一种熟悉就使用哪一种

1. **1.2.OgnlContext对象（了解）**

OgnlContext对象是ognl表达式语言的核心。  
但是项目中不会要求写OgnlContext的代码，Ognl标签其实是调用了OgnlContext对象。所以只做了解即可。

OgnlContext对象在源码中实现了Map接口：  
public class OgnlContext implements Map {……}

Ognl表达式语言取值，也是用java代码取值的，原理就是使用OgnlContext和Ognl这两个类，**只需要记住，Ognl取根元素不用#号，取非根元素要使用#号**：

**OgnlContext类**  
硬编码方式，了解OgnlContext对象，因为OgnlContext对象实现是Map接口，所有OgnlContext本质就是一个Map，可以使用map方法：

OgnlContext context = new OgnlContext();  
context.put("uesr",user);  
context.put("address",address);  
context.setRoot(address);

**Ognl类**  
Ognl类也是Ognl底层运行的代码，常用的api如下：

Object obj1 = Ognl.parseExpression(“country”); 解析ognl表达式  
Ognl.getValue(obj1, context, context.getRoot()); 获取ognl的表达式值,obj1是上面一个api，其他两个分别是创建的上下文对象以及一个不用修改的参数  
Object obj2 = Ognl.parseExpression(“language.toUpperCase()”); 方法调用  
Object obj3 = Ognl.parseExpression("@java.lang.Integer@toBinaryString(10)");等同于上面  
Object obj4 = Ognl.parseExpression(“@@min(10,4)”); Math类的方法直接调用,静态方法的调用

代码示例如下：

package o\_ognl;

import ognl.Ognl;

import ognl.OgnlContext;

import ognl.OgnlException;

import org.junit.Test;

/\*\*

\* OgnlContext用法

\* 1.使用Ognl表达式语言取值，如果取非根元素的值，必须用#号

\* 2.使用Ognl表达式语言取值，如果取根元素的值，不用#号

\* 3.Ognl可以调用静态方法

\*/

public class OgnlDemo {

//非根元素

@Test

public void testOgnl1() throws OgnlException {

//创建一个Ognl上下文对象

OgnlContext context = new OgnlContext();

/\*\*

\* 1.OgnlContext放入基本变量数据

\*/

//放入数据

context.put("cn","China");

//获取数据（map）

String value = (String)context.get("cn");

System.out.println(value);

/\*\*

\* 2.OgnlContext放入对象数据

\*/

//创建对象，设置对象属性

User user = new User();

user.setId(100);

user.setName("Jack");

//【往非根元素放入数据，取值的时候表达式要用“#”】

context.put("user",user);

//获取对象属性

//使用这种方式也可以获取

Object s = context.get("user");

System.out.println(s);

//使用Ognl表达式来获取

//举例：例如标签<s:a value="#user.id">取值，实际上就是运行了下面的代码获取的

//先构建一个Ognl表达式，再解析表达式

Object ognl = Ognl.parseExpression("#user.id");//构建Ognl表达式

Object value1 = Ognl.getValue(ognl, context, context.getRoot());//解析表达式

System.out.println(value1);

User user1 = new User();

user1.setId(100);

user1.setName("Jack");

context.setRoot(user1);

Object ognl1 = Ognl.parseExpression("id");//构建Ognl表达式

Object value2 = Ognl.getValue(ognl1, context, context.getRoot());//解析表达式

System.out.println(value2);

}

//根元素，

@Test

public void testOgnl2() throws OgnlException {

OgnlContext context = new OgnlContext();

User user1 = new User();

user1.setId(100);

user1.setName("Jack");

context.setRoot(user1);

//根元素直接使用id,不需要加#号

Object ognl1 = Ognl.parseExpression("id");//构建Ognl表达式

Object value2 = Ognl.getValue(ognl1, context, context.getRoot());//解析表达式

System.out.println(value2);

}

//ognl对静态方法调用的支持

@Test

public void testOgnl3() throws Exception{

//创建一个Ognl上下文对象

OgnlContext context = new OgnlContext();

//Ognl表达式语言，调用类的静态方法

// Object ognl = Ognl.parseExpression("@Math@floor(10.9)");

//由于Math类在开发中比较常用，所有也可以这样写

Object ognl = Ognl.parseExpression("@@floor(10.9)");

Object value = Ognl.getValue(ognl, context, context.getRoot());

System.out.println(value);

}

}

1. **1.3.ValueStack对象**

**1.ValueStack即值栈对象**  
ValueStack实际是一个接口，在Struts2中利用Ognl时，实际上使用的是实现了该接口的OgnlValueStack类，这个类是Struts2利用Ognl的基础

**2.ValueStack特点**  
ValueStack贯穿整个Action的生命周期（每个Action类的对象实例都拥有一个ValueStack对象），即用户每次访问struts的action，都会创建一个Action对象、值栈对象、ActionContext对象，然后把Action对象放入值栈中；最后再把值栈对象放入request中，传入jsp页面。相当于**一个数据的中转站**，在其中保存当前Action对象和其他相关对象。**Struts2框架把ValueStack对象保存在名为“struts。valueStack”的request请求属性中。**

**3.ValueStack存储对象**

代码调试的时候，发现有一个root是compundRoot类，继承ArrayList，保存的是action对象；还有一个OgnlContext是继承Map，保存数据。  
所以ValueStack存储对象时是分两个地方来存的，也即ValueStack对象的组成是由List栈和Map栈构成的：  
ObjectStack：Struts把根元素，即action对象及全局属性存入ObjectStack中---List

list栈主要存储：action对象，Map对象（通过vs.set()设置），通过push方法设置的对象，以及其他代理对象

根元素的存储示例：

//存储值栈对象

ActionContext ac = ActionContext.getContext();

ValueStack vs = ac.getValueStack();

vs.set("user1",new User(100,"Jack1"));//Map

vs.push(new User(100,"Jack2"));//栈顶

ContextMap：Struts把各种各样的映射关系（域数据）存入ContextMap中  
**Struts会把下面这些映射存入ContextMap中：**

* parameter：该Map中包含当前请求的请求参数
* request：该Map中包含当前request对象中的所有属性
* Session：该Map中包含当前Session对象中的所有属性
* application：该Map中包含当前application对象中的所有属性
* attr：该Map按如下顺序来检索某个属性：request，Session，application  
  非根元素Map中存放数据的方法示例：

//存储值栈对象

ActionContext ac = ActionContext.getContext();

//映射数据

ac.getContextMap().put("request\_data", "request\_data");

ac.getSession().put("session\_data", "session\_data");

ac.getApplication().put("application\_data","application\_data");

从栈中取值的两种方式：

//一、获取值栈对象的两种方式，是等价的

public void getVs() {

//获取值栈对象，方式1：

HttpServletRequest request = ServletActionContext.getRequest();

ValueStack vs1 = (ValueStack) request.getAttribute("struts.valueStack");

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//获取值栈对象，方式2:

ActionContext ac = ActionContext.getContext();

ValueStack vs2 = ac.getValueStack();

System.out.println(vs1==vs2);//true

}

在jsp页面中，对不同ValueStack中的不同类型取值方法不同，  
如果是根元素取值，直接写表达式；  
非根元素(request，Session，application，att，parmeters)必须用#号，例#request.cn

1. **1.4.JSP页面中获取ValueStack数据**

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

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

<html>

<head>

<title>jsp页面取值</title>

</head>

<body>

index页面

<%--页面，必须要拿到ValueStack--%>

<%--struts的调试标签，可以观测值栈数据--%>

<s:debug/>

<br/>1.取根元素的值<br/>

<s:property value="user.id"/>

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

<br/>2.取非根元素<br/>

<s:property value="#request.cn"/>

<s:property value="#request.request\_data"/>

<s:property value="#session.session\_data"/>

<s:property value="#application.application\_data"/><br/>

<%--attr按顺序自动找request/session/application,找到后立刻返回--%>

<s:property value="#attr.application\_data"/>

<%--获取请求参数数据--%>

<s:property value="#parameters.userName"/>

</body>

</html>

1. **2.Struts标签**

struts标签取值，就使用到ognl表达式语言。

1. **2.1.Ognl使用方式**

1.4.中使用的就是Ognl表达式取值。使用方式是：  
1.引入<%@taglib prefix="s" uri="/struts-tags" %>  
2.使用 <s:property value="user.name"/>标签获取取值，取值的时候要注意根元素（全局变量）不用#号，其他的都用#号

还有一些标签需要学习：

1. **2.2.迭代标签**

**Iterator：**标签用于对集合进行迭代，这里的集合包含List，Set和数组

* value:可选属性，指定被迭代的集合，如果没有设置该属性，则使用ValueStack栈顶的集合。
* var：可选属性，引用变量的名称
* status：可选属性，该属性指定迭代是的IteratorStatus实例，该实例包含一下一个方法;

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 isEven(),返回当前迭代元素的索引是否是偶数  
boolean isOdd(),返回当前迭代元素的索引是否是奇数  
boolean isFirst(),返回当前迭代元素是否是第一个元素  
boolean isLast(),返回当前迭代元素是否是最后一个元素

代码示例：

<br/>1.list迭代<br/>

<s:iterator var="user" value="#request.list" status="st">

<s:property value="#user.id"/>

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

<s:property value="#st.even"/><br/>

</s:iterator>

<br/>2.map迭代<br/>

<s:iterator var="en" value="#request.map" status="st">

<s:property value="#en.key"/>

<s:property value="#en.value.name"/>

</s:iterator>

1. **2.3.Ognl动态建立集合**

<%--Ognl表达式可以取值，也可以构建动态集合--%>

<br/>1.构建list集合<br/>

<s:iterator var="str" value="{'a','b','c'}">

<s:property value="#str"/>

</s:iterator>

<br/>2.构建map集合<br/>

<s:iterator var="en" value="#{'cn':'China','usa':'America'}">

<s:property value="en.key"/>

<s:property value="en.value"/><br/>

</s:iterator>

1. **2.4.简单UI标签**

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

<%--服务器标签:最终被解析为html标签--%>

<s:form action="/user\_login" method="POST" name="frmLogin">

<s:textfield name="user.userName" label="用户名"/>

<s:textfield name="user.pwd" label="密码"/>

<s:submit value="登录"/>

</s:form>

也可以给form指定主题，方法如下：

<!-- 服务器标签 : 最终别解析为html标签-->

<s:form action="/user\_login" method="post" name="frmLogin" id="frmLogin" theme="simple">

用户名：<s:textfield name="user.name"></s:textfield>

密码：<s:textfield name="user.pwd"></s:textfield>

<s:submit value="登陆"></s:submit>

</s:form>

**注意：**  
给form指定主题，form下所有的表单元素都应用此主题

对Struts标签默认的主题样式：default.xml/struts.ui.theme=xhtml  
可以通过常量修改，改为简单主题：

<!-- 修改主题 (当前项目所有的标签都用此主题)-->

<constant

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

1. **2.5.Ognl表达式语言几个符号**
2. **：获取非根元素，动态创建map集合**

$：配置文件取值  
%：提供一个ognl表达式运行环境

<body>

<br/>获取request域数据<br/>

<!-- property 标签是对象类型的标签，默认支持ognl表达式, 会从根元素去China名称对应的值 -->

<s:property value="China"/> <br/>

<!-- 如果直接赋值，需要用单引号 -->

<s:property value="'China'"/> <br/>

<s:property value="%{#request.cn}"/> <br/>

<!-- 值类型的标签，value值默认就是值类型，不支持ognl表达式 -->

国家：<s:textfield name="txtCountry" value="%{#request.cn}"></s:textfield>

</body>