**总结OGNL的使用方法：**

**访问属性**

|  |
| --- |
| 名字属性获取:<s:property value="user.username"/><br>  地址属性获取:<s:property value="user.address.addr"/><br> |

**访问方法**

|  |
| --- |
| 调用值栈中对象的普通方法：<s:property value="user.get()"/><br> |

**访问静态属性和方法**

|  |
| --- |
| 调用Action中的静态方法：<s:property value="@struts.action.LoginAction@get()"/>  调用JDK中的类的静态方法：<s:property value="@[**Java**](http://lib.csdn.net/base/java).lang.Math@floor(44.56)"/><br>  调用JDK中的类的静态方法(同上)：<s:property value="@@floor(44.56)"/><br>  调用JDK中的类的静态方法：<s:property value="@java.util.Calendar@getInstance()"/><br>  调用普通类中的静态属性：<s:property value="@struts.vo.Address@TIPS"/><br> |

**访问构造方法**

|  |
| --- |
| 调用普通类的构造方法:<s:property value="new struts.vo.Student('李晓红' , '美女' , 3 , 25).username"/> |

**1.5. 访问数组**

|  |
| --- |
| 获取List:<s:property value="testList"/><br>  获取List中的某一个元素(可以使用类似于数组中的下标获取List中的内容):  <s:property value="testList[0]"/><br>  获取Set:<s:property value="testSet"/><br>  获取Set中的某一个元素(Set由于没有顺序，所以不能使用下标获取数据):  <s:property value="testSet[0]"/><br> ×  获取Map:<s:property value="testMap"/><br>  获取Map中所有的键:<s:property value="testMap.keys"/><br>  获取Map中所有的值:<s:property value="testMap.values"/><br>  获取Map中的某一个元素(可以使用类似于数组中的下标获取List中的内容):  <s:property value="testMap['m1']"/><br>  获取List的大小:<s:property value="testSet.size"/><br> |

**访问集合 – 投影、选择(? ^ $)**

|  |
| --- |
| 利用选择获取List中成绩及格的对象:  <s:property value="stus.{?#this.grade>=60}"/><br>  利用选择获取List中成绩及格的对象的username:  <s:property value="stus.{?#this.grade>=60}.{username}"/><br>  利用选择获取List中成绩及格的第一个对象的username:  <s:property value="stus.{?#this.grade>=60}.{username}[0]"/><br>  利用选择获取List中成绩及格的第一个对象的username:  <s:property value="stus.{^#this.grade>=60}.{username}"/><br>  利用选择获取List中成绩及格的最后一个对象的username:  <s:property value="stus.{$#this.grade>=60}.{username}"/><br>  利用选择获取List中成绩及格的第一个对象然后求大小:  <s:property value="stus.{^#this.grade>=600}.{username}.size"/><br> |

**集合的伪属性**

OGNL能够引用集合的一些特殊的属性,这些属性并不是JavaBeans模式,例如size(),length()等等. 当表达式引用这些属性时,OGNL会调用相应的方法,这就是伪属性.

|  |  |
| --- | --- |
| 集合 | 伪属性 |
| Collection(inherited by Map, List & Set) | size ,isEmpty |
| List | iterator |
| Map | keys , values |
| Set | iterator |
| Iterator | next , hasNext |
| Enumeration | next , hasNext , nextElement , hasMoreElements |

**Lambda   :[…]**

格式：:[…]

|  |
| --- |
| 使用Lambda表达式计算阶乘:  <s:property value="#f = :[#this==1?1:#this\*#f(#this-1)] , #f(4)"/><br> |

**OGNL中#的使用**

#可以取出堆栈上下文中的存放的对象.

|  |  |  |
| --- | --- | --- |
| 名称 | 作用 | 例子 |
| parameters | 包含当前HTTP请求参数的Map | #parameters.id[0]作用相当于  request.getParameter("id") |
| request | 包含当前HttpServletRequest的属性（attribute)的Map | #request.userName相当于  request.getAttribute("userName") |
| session | 包含当前HttpSession的属性（attribute）的Map | #session.userName相当于  session.getAttribute("userName") |
| application | 包含当前应用的ServletContext的属性（attribute）的Map | #application.userName相当于  application.getAttribute("userName") |
| attr | 用于按request > session > application顺序访问其属性（attribute） |  |

获取Paraments对象的属性：<s:property value="#parameters.username"/>

**OGNL中%的使用**

用%{}可以取出存在值堆栈中的Action对象,直接调用它的方法.

例如你的Action如果继承了ActionSupport .那么在页面标签中，用%{getText('key')}的方式可以拿出国际化信息.

 OGNL中$的使用

“$”有两个主要的用途

l         用于在国际化资源文件中，引用OGNL表达式

l         在Struts 2配置文件中，引用OGNL表达式

**值栈**

ValueStack对象。这个对象贯穿整个Action的生命周期（每个Action类的对象实例会拥有一个ValueStack对象）。当Struts 2接收到一个.action的请求后，会先建立Action类的对象实例，但并不会调用Action方法，而是先将Action类的相应属性放到ValueStack对象的顶层节点（ValueStack对象相当于一个栈）。

在Action中获得ValueStack对象：ActionContext.getContext().getValueStack()

l         Top语法

使用Top获取值栈中的第二个对象:<s:property value="[1].top.对象"/>

l         N语法

使用N获取值栈中的第二个对象:<s:property value="[1].对象"/>

l         @语法

调用action中的静态方法：<s:property value="@vs1@静态方法"/> vs：值栈 1：表示第一个。