**一．OGNL中的#、%和$符号**

      #、%和$符号在OGNL表达式中经常出现，而这三种符号也是开发者不容易掌握和理解的部分。在这里我们简单介绍它们的相应用途。

**1．#符号的三种用法**

   1）访问非根对象属性，例如示例中的#session.msg表达式，由于Struts 2中值栈被视为根对象，所以访问其他非根对象时，需要加#前缀。实际上，#相当于ActionContext. getContext()；#session.msg表达式相当于ActionContext.getContext().getSession(). getAttribute("msg") 。

   2）用于过滤和投影（projecting）集合，如示例中的persons.{?#this.age>20}。

   3） 用来构造Map，例如示例中的#{'foo1':'bar1', 'foo2':'bar2'}。

**2．%符号**

      %符号的用途是在标志的属性为字符串类型时，计算OGNL表达式的值。如下面的代码所示：

<h3>构造Map</h3>

    <s:set name="foobar" value="#{'foo1':'bar1', 'foo2':'bar2'}" />

    <p>The value of key "foo1" is <s:property value="#foobar['foo1']" /></p>

    <p>不使用％：<s:url value="#foobar['foo1']" /></p>

   <p>使用％：<s:url value="%{#foobar['foo1']}" /></p>

运行界面如下所示。

he value of key "foo1" is bar1

不使用%：#foobar['foo1']

使用%：bar1

**3．$符号**

$符号主要有两个方面的用途。

     1） 在国际化资源文件中，引用OGNL表达式，例如国际化资源文件中的代码：reg.agerange=国际化资源信息：年龄必须在${min}同${max}之间。

     2） 在Struts 2框架的配置文件中引用OGNL表达式，例如下面的代码片断所示：

<validators>  
  
    <field name="intb">  
  
            <field-validator type="int">  
  
            <param name="min">10</param>  
  
            <param name="max">100</param>  
  
            <message>BAction-test校验：数字必须为${min}为${max}之间！</message>  
  
        </field-validator>  
  
    </field>  
  
</validators>  
  
  
**二．我们一起看一下OGNL常用表达式：**

1. 当使用OGNL调用静态方法的时候，需要按照如下语法编写表达式：

@package.classname@methodname(parameter)

2. 对于OGNL来说，java.lang.Math是其的默认类，如果调用java.lang.Math的静态方法时，无需指定类的名字，比如：@@min(4, 10);

3. 对于OGNL来说，数组与集合是一样的，都是通过下标索引来去访问的。  
  
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>  
4. 使用OGNL来处理映射（Map）的语法格式如下所示：

#{‘key1’: ‘value1’, ‘key2’: ‘value2’, ‘key3’: ‘value3’};

5. 过滤（filtering）：collection.{? expression}

6. OGNL针对集合提供了一些伪属性（如size，isEmpty），让我们可以通过属性的方式来调用方法（本质原因在于集合当中的很多方法并不符合JavaBean的命名规则），但我么你依然还可以通过调用方法来实现与伪属性相同的目的。

7. 过滤（filtering），获取到集合中的第一个元素：collection.{^ expression}

8. 过滤（filtering），获取到集合中的最后一个元素：collection.{& expression}

9. 在使用过滤操作时，我们通常都会使用#this，该表达式用于代表当前正在迭代的集合中的对象（联想增强的for循环）

10. 投影（projection）：collection.{expression}

11. 过滤与投影之间的差别：类比于数据库中的表，过滤是取行的操作，而投影是取列的操作。 具体举例如下：

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>

12． 在Struts2中，根对象就是ValueStack。在Struts2的任何流程当中，ValueStack中的最顶层对象一定是Action对象。

13. parameters，#parameters.username

request, #request.username

session, #session.username

application, #application.username

attr, #attr.username

以上几个对象叫做“命名对象”。

14. 访问静态方法或是静态成员变量的改进。

@vs@method

15. 关于Struts2标签库属性值的%与#号的关系：

1）. 如果标签的属性值是OGNL表达式，那么无需加上%{}。

2）. 如果标签的属性值是字符串类型，那么在字符串当中凡是出现的%{}都会被解析成OGNL表达式，解析完毕后再与其他的字符串进行拼接构造出最后的字符串值。

3）. 我们可以在所有的属性值上加%{}，这样如果该属性值是OGNL表达式，那么标签处理类就会将%{}忽略掉。

最后一起用代码说话，简单的看一下ognl操作的示例：

**1）上下文环境中使用OGNL**

**public static void main(String[] args)  
    {  
         
        Map<String , Object> context = new HashMap<String , Object>();  
  
        Person person1 = new Person();  
        person1.setName("zhangsan");  
        
        Person person2 = new Person();  
        person2.setName("lisi");  
  
        Person person3 = new Person();  
        person3.setName("wangwu");  
  
         
        Person person4 = new Person();  
        person4.setName("zhaoliu");  
  
         
        context.put("person1", person1);  
        context.put("person2", person2);  
        context.put("person3", person3);  
  
        try  
        {  
             
            Object value = Ognl.getValue("name", context, person2);  
            System.out.println("ognl expression \"name\" evaluation is : " + value);  
  
             
            Object value2 = Ognl.getValue("#person2.name", context, person2);  
            System.out.println("ognl expression \"#person2.name\" evaluation is : " + value2);  
  
             
            Object value3 = Ognl.getValue("#person1.name", context, person2);  
            System.out.println("ognl expression \"#person1.name\" evaluation is : " + value3);  
  
             
            Object value4 = Ognl.getValue("name", context, person4);  
            System.out.println("ognl expression \"name\" evaluation is : " + value4);  
  
             
            Object value5 = Ognl.getValue("#person4.name", context, person4);  
            System.out.println("ognl expression \"person4.name\" evaluation is : " + value5);  
  
             
            // Object value6 = Ognl.getValue("#person4.name", context, person2);  
           // System.out.println("ognl expression \"#person4.name\" evaluation is : " + value6);  
  
        }**  
**2）使用OGNL调用方法**

**public static void main(String[] args)  
    {  
         
        OgnlContext context = new OgnlContext();  
  
        People people1 = new People();  
        people1.setName("zhangsan");  
  
        People people2 = new People();  
        people2.setName("lisi");  
  
        People people3 = new People();  
        people3.setName("wangwu");  
  
        context.put("people1", people1);  
        context.put("people2", people2);  
        context.put("people3", people3);  
        
        context.setRoot(people1);  
  
        try  
        {  
             
            Object value = Ognl.getValue("name.length()", context, context.getRoot());  
            System.out.println("people1 name length is :" + value);  
            
            Object upperCase = Ognl.getValue("#people2.name.toUpperCase()", context, context.getRoot());  
            System.out.println("people2 name upperCase is :" + upperCase);  
  
            Object invokeWithArgs = Ognl.getValue("name.charAt(5)", context, context.getRoot());  
            System.out.println("people1 name.charAt(5) is :" + invokeWithArgs);  
  
             
            Object min = Ognl.getValue("@java.lang.Math@min(4,10)", context, context.getRoot());  
            System.out.println("min(4,10) is :" + min);  
  
             
            Object e = Ognl.getValue("@java.lang.Math@E", context, context.getRoot());  
            System.out.println("E is :" + e);  
        }  
  
3）使用OGNL操作集合**

**public static void main(String[] args) throws Exception  
    {  
        OgnlContext context = new OgnlContext();  
        
        Classroom classroom = new Classroom();  
        classroom.getStudents().add("zhangsan");  
        classroom.getStudents().add("lisi");  
        classroom.getStudents().add("wangwu");  
        classroom.getStudents().add("zhaoliu");  
        classroom.getStudents().add("qianqi");  
        
        Student student = new Student();  
        student.getContactWays().put("homeNumber", "110");  
        student.getContactWays().put("companyNumber", "119");  
        student.getContactWays().put("mobilePhone", "112");  
        
        context.put("classroom", classroom);  
        context.put("student", student);  
        context.setRoot(classroom);  
  
         
        Object collection = Ognl.getValue("students", context, context.getRoot());  
        System.out.println("students collection is ：" + collection);  
  
         
        Object firstStudent = Ognl.getValue("students[0]", context, context.getRoot());  
        System.out.println("first student is : " + firstStudent);  
  
         
        Object size = Ognl.getValue("students.size()", context, context.getRoot());  
        System.out.println("students collection size is :" + size);  
  
        System.out.println("--------------------------飘逸的分割线--------------------------");  
        
        Object mapCollection = Ognl.getValue("#student.contactWays", context, context.getRoot());  
        System.out.println("mapCollection is :" + mapCollection);  
  
        Object firstElement = Ognl.getValue("#student.contactWays['homeNumber']", context, context.getRoot());  
        System.out.println("the first element of contactWays is :" + firstElement);  
  
        System.out.println("--------------------------飘逸的分割线--------------------------");  
  
         
        Object createCollection = Ognl.getValue("{'aa','bb','cc','dd'}", context, context.getRoot());  
        System.out.println(createCollection);  
  
         
        Object createMapCollection = Ognl.getValue("#{'key1':'value1','key2':'value2'}", context, context.getRoot());  
        System.out.println(createMapCollection);  
  
    }  
}**