OGNL表达式与值栈

一、OGNL表达式

OGNL 的全称是对象图导航语言（Object-Graph Navigation Language），它是一种功能强大的开源表达式语言，使用这种表达式语言，可以通过某种表达式语法，存取Java对象的任意属性，调用Java对象的方法，同时能够自动实现必要的类型转换。如果把表达式看作是一个带有语义的字符串，那么OGNL无疑成为了这个语义字符串与Java对象之间沟通的桥梁。

1.1 OGNL三要素

1.1.1 表达式

表达式是整个OGNL的核心，OGNL会根据表达式去对象中取值。所有OGNL操作都是针对表达式解析后进行的。它表明了此次OGNL操作要"做什么"。表达式就是一个带有语法含义的字符串，这个字符串规定了操作的类型和操作的内容。OGNL支持大量的表达式语法，不仅支持这种“链式”对象访问路径，还支持在表达式中进行简单的计算。

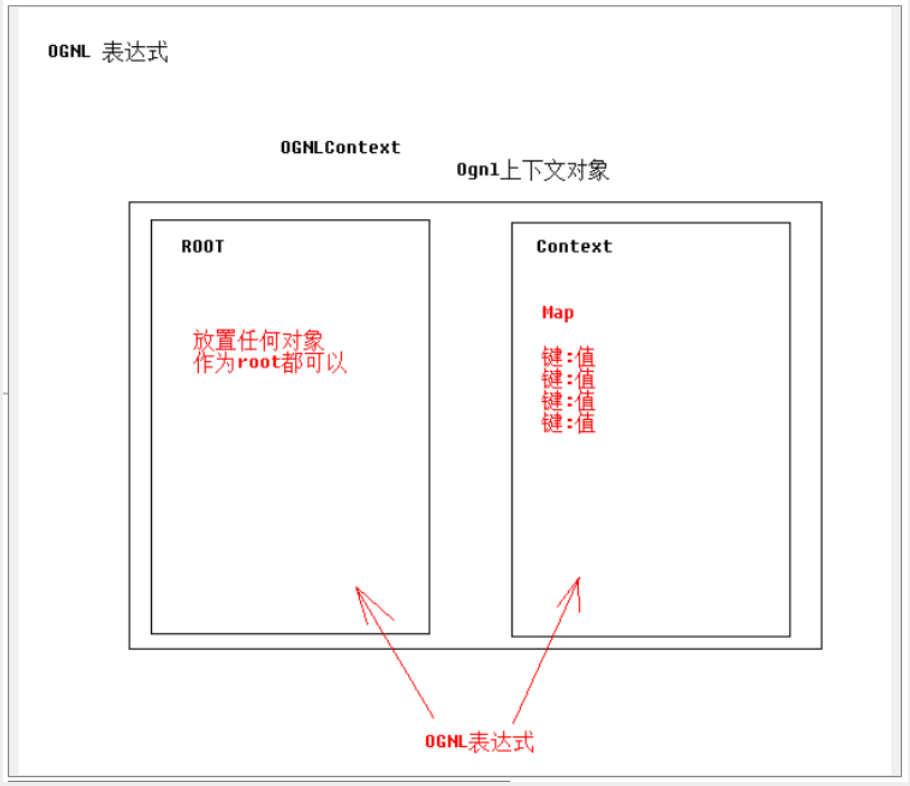
1.1.2 根对象(Root)

Root对象可以理解为OGNL的操作对象，表达式规定了“做什么”，而Root对象则规定了“对谁操作”。OGNL称为对象图导航语言，**所谓对象图，即以任意一个对象为根，通过OGNL可以访问与这个对象关联的其它对象。**

1.1.3 Context对象

实际上OGNL的取值还需要一个上下文环境。设置了Root对象，OGNL可以对Root对象进行取值或写值等操作，Root对象所在环境就是OGNL的上下文环境(Context)。上下文环境规定了OGNL的操作“在哪进行”。上下文环境Context是一个Map类型的对象，在表达式中访问Context中的对象，需

要使用"#"号加上对象名称，即"#对象名称"的形式。



1.2 OGNL的使用

struts2导入的相关包中已经包含了OGNL的包，不用格外导入。

1.2.1 OGNL环境准备

```
14    @Test
15    public void fun1() throws OgnlException{
16        //准备OGNLContext
17            //准备Root
18        User rootUser=new User("tom",18);
19            //准备Context
20        Map<String, User> context=new HashMap<String, User>();
21        context.put("user1", new User("jack",18));
22        context.put("user2", new User("rose",22));
23
24        OgnlContext oc=new OgnlContext();
25        oc.setRoot(rootUser);
26        oc.setValues(context);
27
28    }
```

1.设置Root对象，任何对象都行

2.设置上下文环境，必须是个map集合

3.创建OgnlContext对象并将上述Root对象和上下文环境设置在其中，准备完毕

1.2.2 获取OGNL Root对象中的属性值

```java
//准备OGNLContext
   //准备Root
User rootUser=new User("tom",18);
   //准备Context
Map<String, User> context=new HashMap<String, User>();
context.put("user1", new User("jack",18));
context.put("user2", new User("rose",22));

OgnlContext oc=new OgnlContext();
oc.setRoot(rootUser);
oc.setValues(context);
//书写OGN
//取出root中user对象的name属性
String name = (String) Ognl.getValue("name", oc, oc.getRoot());
Integer age = (Integer) Ognl.getValue("age", oc, oc.getRoot());
System.out.println(name);
System.out.println(age);
```

之后准备OGNL的代码

不再提示

用getValue(" OGNL表达式", OGNLContext对象，root对象)，

可以获取OGNL表达式的值

root对象取值：

只需要把对象的属性名列出即可

其实，root对象，是一个栈的结构，取值每次只能取到栈顶的元素

1.2.3 获取Context对象中的Map集合的属性值

```java
//取出context中键为user1对象的name属性
String name = (String) Ognl.getValue("#user1.name", oc, oc.getRoot());
String name2 = (String) Ognl.getValue("#user2.name", oc, oc.getRoot());
Integer age = (Integer) Ognl.getValue("#user2.age", oc, oc.getRoot());
System.out.println(name);
System.out.println(name2);
System.out.println(age);
```

同样使用getValue表达式得到OGNL表达式的值，只不过context中是Map，不是栈结构，可以取Map中的任意位置值，所以用#对象名获取一个对象

获取对象的属性值不就显而易见了吗

1.2.4 设置Root和Context对象中各属性的值

```java
//修改
//将root和context中的user对象的name属性赋值
String name = (String) Ognl.getValue("name='jerry'", oc, oc.getRoot());
String name2 = (String) Ognl.getValue("#user1.name='郝强勇'", oc, oc.getRoot());
System.out.println(name);
System.out.println(name2);
```

修改就是将原来的属性直接赋值即可

root

上下文环境

1.2.5 利用OGNL表达式调用方法

```java
//调用root中user对象的setName方法
Ognl.getValue("setName('lilei')", oc, oc.getRoot());
String name = (String) Ognl.getValue("getName()", oc, oc.getRoot());

String name2 = (String) Ognl.getValue("#user1.setName('lucy'),#user1.getName()", oc, oc.getRoot());
```

明白了取值，那调用方法就很容易理解了

1.2.6 利用OGNL表达式调用静态方法

```java
//String name = (String) Ognl.getValue("@cn.itheima.a_ognl.HahaUtils@echo('hello 强勇!')", oc, oc.getRoot());
//Double pi = (Double) Ognl.getValue("@java.lang.Math@PI", oc, oc.getRoot());
Double pi = (Double) Ognl.getValue("@@PI", oc, oc.getRoot());
//System.out.println(name);
System.out.println(pi);
```

@全包名@静态方法

1.2.7 利用OGNL表达式创建对象

```
//创建list对象
Integer size = (Integer) Ognl.getValue("{'tom','jerry','jack','rose'}.size()", oc, oc.getRoot());
String name = (String) Ognl.getValue("{'tom','jerry','jack','rose'}[0]", oc, oc.getRoot());
String name2 = (String) Ognl.getValue("{'tom','jerry','jack','rose'}.get(1)", oc, oc.getRoot());

/*System.out.println(size);
System.out.println(name);
System.out.println(name2);*/
//创建Map对象
Integer size2 = (Integer) Ognl.getValue("#{'name':'tom','age':18}.size()", oc, oc.getRoot());
String name3  = (String) Ognl.getValue("#{'name':'tom','age':18}['name']", oc, oc.getRoot());
Integer age   = (Integer) Ognl.getValue("#{'name':'tom','age':18}.get('age')", oc, oc.getRoot());
System.out.println(size2);
System.out.println(name3);
System.out.println(age);
```

二、ognl表达式与struts2结合原理

三、值栈概念和获取值栈

1 之前在 web 阶段，在 servlet 里面进行操作，把数据放到域对象里面，在页面中使用 el 表达式获取到，域对象在一定范围内，存值和取值

2 在 struts2 里面提供本身一种存储机制，类似于域对象，是值栈，可以存值和取值

（1）在 action 里面把数据放到值栈里面，在页面中获取到值栈数据

3 servlet 和 action 区别

（1）Servlet：默认在第一次访问时候创建，创建一次，单实例对象

（2）Action：访问时候创建，每次访问 action 时候，都会创建 action 对象，创建多次，多实例对象

4 值栈存储位置

（1）每次访问 action 时候，都会创建 action 对象，

（2）在每个 action 对象里面都会有一个值栈对象（只有一个）

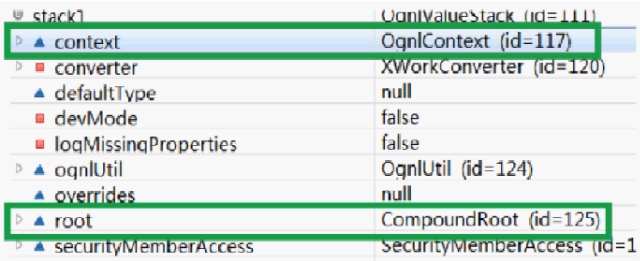
四、值栈对象及其内部结构

每次访问action时都会创建一个对象，每个action中只有一个值栈对象。

```
public class ValueStackAction2 extends ActionSupport {

    public String execute() throws Exception {
        //1.获取ActionContext类的对象
        ActionContext context = ActionContext.getContext();
        //2.调用方法得到值栈对象
        ValueStack valueStack1 = context.getValueStack();
        ValueStack valueStack2 = context.getValueStack();
        System.out.println(valueStack1==valueStack2);
        return SUCCESS;
    }

}
```



context 结构是map集合

```
class OgnlContext extends Object implements Map
```

root 结构是list集合

而且实现了栈的push和pop操作

```
class CompoundRoot extends ArrayList {
```

一般操作都是root里面数据

```
public Object pop() {
    return remove(0);
}

public void push(Object o) {
    add(0, o);
}
```

栈中的两个方法的实现

key 固定	value
request	request对象引用
session	HttpSession对象引用
application	ServletContext对象引用
parameters	传递相关的参数
attr	

* context存储的对象引用

* 三个域对象，向三个域对象放值，名称都相同 setAttribute ("name",value);
* 使用attr操作，获取域对象里面的值，获取域范围最小里面的值

Object	Property Name	Property Value
cn.itheima.b_showvs.Demo1Action	texts	null
	actionErrors	[]
	errors	{}
	fieldErrors	{}
	errorMessages	[]
	container	There is no read method for container
	locale	zh_HANS_CN
	actionMessages	[]
com.opensymphony.xwork2.DefaultTextProvider	texts	null

Stack Context

These items are available using the #key notation

Key	Value
com.opensymphony.xwork2.dispatcher.HttpServletRequest	org.apache.struts2.dispatcher.StrutsRequestWrapper@14f007e
com.opensymphony.xwork2.ActionContext.locale	zh_HANS_CN
com.opensymphony.xwork2.dispatcher.HttpServletResponse	org.apache.catalina.connector.ResponseFacade@1217628
com.opensymphony.xwork2.ActionContext.name	Demo1Action
	{org.apache.tomcat.util.scan.MergedWebXml=<?xml version="1.0" encoding="UTF-8"?> <web-app xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" version="2.5" metadata-complete="true"> <display-name>struts2_day03</display-name> <filter> <filter-name>struts2</filter-name> <filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter</filter-class> <async-supported>false</async-supported> </filter> <filter-mapping> <filter-name>struts2</filter-name> <url-pattern>/*</url-pattern> </filter-mapping> <servlet> <servlet-name>jsp</servlet-name> <servlet-class>org.apache.jasper.servlet.JspServlet</servlet-class> <init-param> <param-name>fork</param-name> <param-value>false</param-value> </init-param> <init-param> <param-name>xpoweredBy</param-name> <param-value>false</param-value> </init-param> <load-on-startup>3</load-on-startup> </servlet> <servlet> <servlet-name>default</servlet-name> <servlet-class>org.apache.catalina.servlets.DefaultServlet</servlet-class> <init-param> <param-name>debug</param-name> <param-value>0</param-value> </init-param> <init-param> <param-name>listings</param-name> <param-value>false</param-value> </init-param> <load-on-startup>1</load-on-startup> </servlet> <servlet-mapping> <servlet-name>jsp</servlet-name> <url-pattern>*.jsp</url-pattern> </servlet-mapping> <servlet-mapping> <servlet-name>jsp</servlet-name> <url-pattern>*.jspx</url-pattern> </servlet-mapping> <servlet-mapping> <servlet-name>default</servlet-name> <url-pattern>/</url-pattern> </servlet-mapping> <session-config> <session-timeout>30</session-timeout> </session-config> <mime-

五、Action向值栈中存数据

5.1 向值栈中放数据的三种方法

5.1.1 set方法

第一种 获取值栈对象，调用值栈对象里面的 set 方法

```
//第一种方式 使用值栈对象里面的 set方法
//1 获取值栈对象
ActionContext context = ActionContext.getContext();
ValueStack stack = context.getValueStack();
//2 调用方法set方法
stack.set("username", "itcastitheima");
```

5.1.2 push方法

```
//3 调用方法push方法
stack.push("abcd");
```

前两种方法会压栈到root空间上，后一种会放到action的属性中。

5.1.3 定义变量并生成get方法即可

```java
public class ValueStackDataAction extends ActionSupport {
    //第三种方法
    private String name;
    private User user;
    private List<User> list=new ArrayList<User>();
    public List<User> getList() {
        return list;
    }
    //2.生成get方法
    public String getName() {
        return name;
    }
    public User getUser() {
        return user;
    }

    public String execute() throws Exception {

        list.add(new User("rose",22));
        list.add(new User("jack",20));
        list.add(new User("luhua",18));
        list.add(new User("wangcai",18));

        user  = new User("tom",20);
        name="ababb";
        return SUCCESS;

    }
}
```

定义变量

生成get方法

赋值

Object	Property Name	Property Value
	texts	null
	actionErrors	[]
	errors	{}
	fieldErrors	{}
	errorMessages	[]
cn.scct.actions.ValueStackDataAction	container	There is no read method for container
	name	ababb
	locale	zh_CN
	actionMessages	[]
	list	[User [name=rose, age=22], User [name=jack, age=20], User [name=luhua, age=18], User [name=wangcai, age=18]]
	user	User [name=tom, age=20]
com.opensymphony.xwork2.DefaultTextProvider	texts	null

5.2 存放字符串、对象和list集合

见5.1.3节即可。

六、结果页面(jsp页面)从值栈中取数据

先在值栈中放数据呗。

```java
public class GetDataAction extends ActionSupport {

    private String username;
    public String getUsername() {
        return username;
    }

    private User user;
                                        上文所说的方法三啊
    public User getUser() {
        return user;
    }

    private List<User> list=new ArrayList<User>();
    public List<User> getList() {
        return list;
    }
    public String execute() throws Exception {
        username="cuihua";
        user=new User("wangcai",20);

        list.add(new User("rose",22));
        list.add(new User("jack",20));
        list.add(new User("luhua",18));
        list.add(new User("wangcai",18));
        return SUCCESS;
    }

}
```

因为放的数据都作为action的添加属性，及其属性值存在，所以用OGNL表达式直接获取即可。

6.1 取字符串

先添加OGNL的标签啊。

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

```jsp
<!-- 1.获取字符串值 -->
<s:property value="username"/><br/>
```

6.2 取对象

```jsp
<!--2. 获取值栈对象的值 -->
<s:property value="user.name"/><br/>
<s:property value="user.age"/><br/>
```

6.3 取list集合

```
<!--3. 获取list中的值 -->
<s:property value="list[0].name"/><br/>
<s:property value="list[0].age"/><br/>
<s:property value="list[1].name"/><br/>
<s:property value="list[1].age"/><br/>
<s:property value="list[2].name"/><br/>
<s:property value="list[2].age"/><br/>
<s:property value="list[3].name"/><br/>
<s:property value="list[3].age"/><br/>
```

方法一

```
<!--使用struts2中的标签iterator-->
<s:iterator value="list">
  <s:property value="name"/>
  <s:property value="age"/>
  <br/>
</s:iterator>
```

方法二

```
<!--
  遍历值栈list集合，得到的每个user对象
  机制：把每次遍历出来的user对象放到context里面
  获取context里面的数据特点：写ognl表达式
 # -->
<s:iterator value="list" var="user">
  <s:property value="#user.name"/>
  <s:property value="#user.age"/>
  <br/>
</s:iterator>
```

方法三

七、其它获取值栈数据的方法(针对set和push的方式)

set和push把它放置于action之上，实时压栈。

```
public String execute() throws Exception {
    //第一种方式，使用值栈对象里面的set方法
    //1.获取值栈对象
    ActionContext context = ActionContext.getContext();
    ValueStack stack = context.getValueStack();
    //2.调用方法set
    stack.set("shanghai", "China");
    //第二种方法 push
    stack.push("ab");
    return SUCCESS;
}
```

```
<s:property value="shanghai"/>
<br/>
<s:property value="[0].top"/>
```

set方法

push方法，此时在栈顶，存top数组，取第一个值，只是写法怪一点

八、#在值栈中的应用

当然是获取ActionContext中的数据了，它是一个map集合，参考OGNL的使用就知道其用法了。
值栈中存了很多域对象，只要在我们获得了某个域对象，并设置了什么参数，就可以用OGNL表达式在页面中获取。
比如我们粗鄙的获取原生request对象后设置值如下：

```
public class GetDataAction3 extends ActionSupport {


    public String execute() throws Exception {
        HttpServletRequest request = ServletActionContext.getRequest();
        request.setAttribute("req", "reqvalue");
        return SUCCESS;
    }

}
```

那么就可以在结果页面中获得：

```
<!-- 获取context里面的数据，写ognl时候，首先添加符号
#context的域对象名称.参数 -->
<s:property value="#request.req"/></br>
```

值栈是用来放数据的，替代域对象。

所以获取原生域对象是不齿的，struts2建议不放在Servlet域对象中，而只推荐放在值栈的root中。