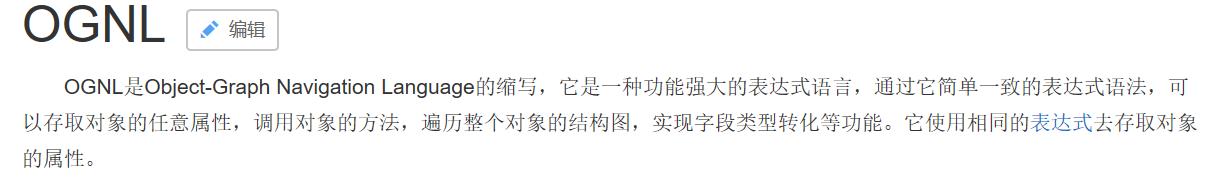
## OGNL表达式概述（了解）





1. OGNL是Object Graphic Navigation Language（对象图导航语言）的缩写

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

通过它简单一致的表达式语法，可以存取对象的任意属性，调用对象的方法，遍历整个对象的结构图，实现字段类型转化等功能。它使用相同的表达式去存取对象的属性

2. Struts2框架使用OGNL作为默认的表达式语言

OGNL是一种比EL强大很多倍的语言

xwork提供 OGNL表达式

ognl-3.0.5.jar

3. OGNL 提供五大类功能

支持对象方法调用

支持类静态的方法调用和值访问

访问OGNL上下文（OGNL context）和ActionContext

支持赋值操作和表达式串联

操作集合对象

4. 测试的代码

// 访问对象的方法

@Test

public void run1() throws OgnlException{

//OGNL的上下文对象

OgnlContext context = new OgnlContext();

// 获取对象的方法

Object obj = Ognl.getValue("'helloworld'.length()", context, context.getRoot());

System.out.println(obj);

}

// 获取OGNL上下文件的对象

@Test

public void run3() throws OgnlException{

OgnlContext context = new OgnlContext();

context.put("name", "haha");

// 获取对象的方法

Object obj = Ognl.getValue("#name", context, context.getRoot());

System.out.println(obj);

}

## 在Struts2框架中使用OGNL表达式

### 1. Struts2引入了OGNL表达式，主要是在JSP页面中获取值栈中的值

### 2. 具体在Struts2中怎么使用呢？如下步骤

需要先引入Struts2的标签库

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

使用Struts2提供的标签中的标签

<s:property value="OGNL表达式"/>

### 3. 在JSP页面使用OGNL表达式

访问对象方法

<s:property value="'hello'.length()"/>

## 值栈的概述

### 问题一：什么是值栈？

值栈就相当于Struts2框架的数据的中转站，向值栈存入一些数据。从值栈中获取到数据。

ValueStack 是 struts2 提供一个接口，实现类 OgnlValueStack ---- 值栈对象 （OGNL是从值栈中获取数据的 ）

Action是多例的，有一起请求，创建Action实例，创建一个ActionContext对象，代表的是Action的上下文对象，还会创建一个ValueStack对象。

每个Action实例都有一个ValueStack对象 （一个请求 对应 一个ValueStack对象 ）

在其中保存当前Action 对象和其他相关对象

Struts 框架把 ValueStack 对象保存在名为 “struts.valueStack” 的请求属性中,request中 （值栈对象 是 request一个属性）

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

### 问题二 ： 值栈的内部结构 ？

值栈由两部分组成

root -- Struts把动作和相关对象压入 ObjectStack 中--List

context -- Struts把各种各样的映射关系(一些 Map 类型的对象) 压入 ContextMap 中

Struts会默认把下面这些映射压入ContextMap（context）中

注意：request代表的是Map集合的key值，value的值其实也是一个Map集合。

parameters: 该 Map 中包含当前请求的请求参数 ?name=jack&password=123

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

ValueStack中 存在root属性 (CompoundRoot) 、 context 属性 （OgnlContext ）

CompoundRoot 就是ArrayList

OgnlContext 就是 Map

context 对应Map 引入 root对象

context中还存在 request、 session、application、 attr、 parameters 对象引用

OGNL表达式访问值栈中的数据

访问root中数据时 不需要 #

访问 request、 session、application、 attr、 parameters 对象数据 必须写 #

操作值栈 默认指 操作 root 元素

### 问题三 ： 值栈对象的创建，ValueStack 和 ActionContext 是什么关系？

值栈对象是请求时创建的

ActionContext是绑定到当前的线程上，那么在每个拦截器或者Action中获取到的ActionContext是同一个。

ActionContext中存在一个Map集合，该Map集合和ValueStack的context是同一个地址。

ActionContext中可以获取到ValueStack的引用，以后再开发，使用ActionContext来获取到值栈对象

### 问题四 ： 如何获得值栈对象

获得值栈对象 有三种方法

ValueStack vs1 = (ValueStack) ServletActionContext.getRequest().getAttribute("struts.valueStack");

ValueStack vs2 = (ValueStack) ServletActionContext.getRequest().getAttribute(ServletActionContext.STRUTS\_VALUESTACK\_KEY);

ValueStack vs3 = ActionContext.getContext().getValueStack();

### 问题五： 向值栈保存数据 （主要针对root栈）

值栈里保存数据的两种方式：

1. struts2默认会吧action类的实例压到contextMap栈里， 该类的属性也都全部在里面。
2. 自己在方法里，手动获得值栈对象，自己调用下面的方法来存值

valueStack.push(Object obj);

push方法的底层调用root对象的push方法（把元素添加到0位置）

valueStack.set(String key, Object obj);

源码获取map集合（map有可能是已经存在的，有可能是新创建的），把map集合push到栈顶，再把数据存入到map集合中。

在jsp中 通过 <s:debug /> 查看值栈的内容

### 问题六： 在JSP中获取值栈的数据

总结几个小问题：

访问root中数据 不需要#

访问context其它对象数据 加 #

如果向root中存入对象的话，优先使用push方法。

如果向root中存入集合的话，优先要使用set方法。

在OgnlContext中获取数据

在Action中向域对象中存入值

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

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

application:<s:property value="#application.username"/>

attr:<s:property value="#attr.username"/>

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

代码如下

<!--

// vs.push("哈哈");

// 获取到栈顶的值

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

-->

<!--

// 栈顶是map集合，通过key获取值

vs.set("msg", "小妹");

<s:property value="[0].top.msg"/>

-->

<!--

vs.push(user);

// 栈顶放user对象

<s:property value="[0].top.username"/>

<s:property value="[0].top.password"/>

// [0].top 关键字是可以省略的 findValue()

<s:property value="username"/>

-->

<!--

vs.set("user", user);

<s:property value="[0].top.user.username"/>

<s:property value="[0].top.user.password"/>

// 省略关键字

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

-->

<!--

// 在ValueStack1Action提供了成员的属性

private User user = new User("小泽","456");

public User getUser() {

return user;

}

public void setUser(User user) {

this.user = user;

}

User user = new User("小苍","123");

vs.set("user", user);

// 从栈顶开始查找，找user的属性，显示名称 返回的小苍

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

// [1].top获取ValueStack1Action [1].top.user返回user对象 [1].top.user.username获取对象的属性名称

<s:property value="[1].top.user.username"/>

-->

<!--

栈顶是list集合

vs.push(ulist);

<s:property value="[0].top[0].username"/>

<s:property value="[0].top[1].username"/>

-->

<!--

vs.set("ulist", ulist);

<s:property value="ulist[0].username"/>

-->

<!-- 迭代的标签

属性

\* value 要迭代的集合，需要从值栈中获取

\* var 迭代过程中，遍历的对象

\* var编写上，把迭代产生的对象默认压入到context栈中，从context栈取值，加#号

\* var不编写，默认把迭代产生的对象压入到root栈中

for(User user:ulist){}

// 编写var的属性 默认会把当前元素压到context对象

<s:iterator value="ulist" var="u">

<s:property value="#u.username"/>

<s:property value="#u.password"/>

</s:iterator>

// 没有编写var关键字 默认会吧当前对象压到root对象

<s:iterator value="ulist">

<s:property value="username"/>

<s:property value="password"/>

</s:iterator>

-->

<!-- 从context栈中获取值，加#号

HttpServletRequest request = ServletActionContext.getRequest();

request.setAttribute("msg", "美美");

request.getSession().setAttribute("msg", "小风");

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

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

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

<s:property value="#attr.msg"/>

-->

<!-- 在JSP页面上，查看值栈的内部结构 -->

<s:debug></s:debug>

EL表达式也会获取到值栈中的数据

### 问题七：为什么EL也能访问值栈中的数据？

StrutsPreparedAndExecuteFilter的doFilter代码中 request = prepare.wrapRequest(request);

对Request对象进行了包装 ，StrutsRequestWrapper

增强了request的 getAttribute

Object attribute = super.getAttribute(s);

if (attribute == null) {

attribute = stack.findValue(s);

}

访问request范围的数据时，如果数据找不到，去值栈中找

request对象 具备访问值栈数据的能力 （查找root的数据）

## 总结OGNL表达式的特殊的符号

## # 符号的用法

\* 获得contextMap中的数据

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

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

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

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

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

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

\* 构建一个map集合

\* 例如：

<s:radio name="sex" list="{'男','女'}"></s:radio>

<s:radio name="sex" list="#{'0':'男','1':'女'}"></s:radio>

## % 符号的用法

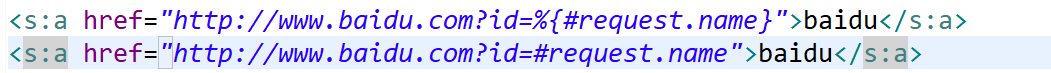
\* 强制字符串解析成OGNL表达式。

例如：在request域中存入值，然后在文本框（<s:textfield>）中取值，现在到value上。

<s:textfield value="%{#request.msg}"/>

\* { }中值用''引起来,此时不再是ognl表达式,而是普通的字符串

例如：<s:property value="%{'#request.msg'}"/>



### $ 符号的用法

\* 在配置文件中可以使用OGNL表达式，例如：文件下载的配置文件。

<action name="download1" class="cn.qf.demo2.DownloadAction">

<result name="success" type="stream">

<param name="contentType">${contentType}</param>

<param name="contentDisposition">attachment;filename=${downFilename}</param>

</result>

</action>