OGNL表达式语言

OGNL表达式语言

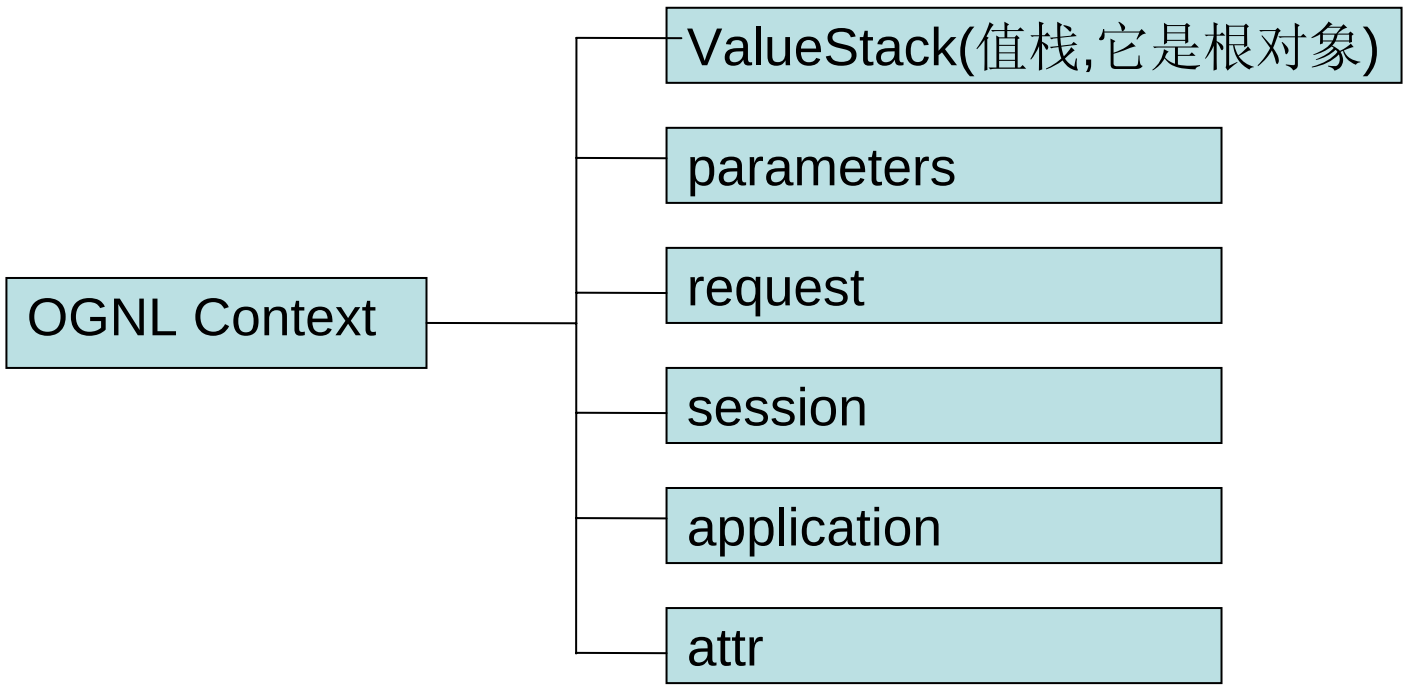
- OGNL是Object Graphic Navigation Language（对象图导航语言）的缩写，它是一个开源项目。 Struts 2框架使用OGNL作为默认的表达式语言。

- 相对EL表达式，它提供了平时我们需要的一些功能，如：
- 支持对象方法调用，如xxx.sayHello()；
- 支持类静态方法调用和值访问，表达式的格式为@[类全名（包括包路径）]@[方法名 | 值名]，例如：@java.lang.String@format('foo %s', 'bar')
- 操作集合对象。

- 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接口，在Struts2中上下文（Context）的实现为ActionContext，下面是上下文（Context）的结构示意图

OGNL表达式语言

Struts 2中的OGNL Context实现者为ActionContext，它结构示意图如下:



当Struts2接受一个请求时，会迅速创建ActionContext，ValueStack，action 。然后把action存放进ValueStack，所以action的实例变量可以被OGNL访问。

OGNL表达式语言

- 访问上下文（Context）中的对象需要使用#符号标注命名空间，如#application、#session
- 另外OGNL会设定一个根对象（root对象），在Struts2中根对象就是ValueStack（值栈）。如果要访问根对象（即ValueStack）中对象的属性，则可以省略#命名空间，直接访问该对象的属性即可。
- 在struts2中，根对象ValueStack的实现类为OgnlValueStack，该对象不是我们想像的只存放单个值，而是存放一组对象。在OgnlValueStack类里有一个List类型的root变量，就是使用他存放一组对象
- |--request
- |--application
- context ------|--OgnlValueStack root变量[action, OgnlUtil, ...]
- |--session
- |--attr
- |--parameters
- 在root变量中处于第一位的对象叫栈顶对象。通常我们在OGNL表达式里直接写上属性的名称即可访问root变量里对象的属性，搜索顺序是从栈顶对象开始寻找，如果栈顶对象不存在该属性，就会从第二个对象寻找，如果没有找到就从第三个对象寻找，依次往下访问，直到找到为止。
- 大家注意： Struts2中，OGNL表达式需要配合Struts标签才可以使用。如：<s:property value="name"/>

OGNL表达式语言

- 由于ValueStack(值栈)是Struts 2中OGNL的根对象，如果用户需要访问值栈中的对象，在JSP页面可以直接通过下面的EL表达式访问ValueStack(值栈)中对象的属性：
- ${foo} //获得值栈中某个对象的foo属性

- 如果访问其他Context中的对象，由于他们不是根对象，所以在访问时，需要添加#前缀。
- **application对象**：用于访问ServletContext，例如#application.userName或者#application['userName']，相当于调用ServletContext的getAttribute("username")。

- **session对象**：用来访问HttpSession，例如#session.userName或者#session['userName']，相当于调用session.getAttribute("userName")。

- **request对象**：用来访问HttpServletRequest属性（attribute）的Map，例如#request.userName或者#request['userName']，相当于调用request.getAttribute("userName")。

- **parameters对象**：用于访问HTTP的请求参数，例如#parameters.userName或者#parameters['userName']，相当于调用request.getParameter("username")。

- **attr对象**：用于按page->request->session->application顺序访问其属性。

采用OGNL表达式创建List/Map集合对象

- 如果需要一个集合元素的时候（例如List对象或者Map对象），可以使用OGNL中同集合相关的表达式。
- 使用如下代码直接生成一个List对象：
- <s:set name="list" value="{'zhangming','xiaoi','liming'}" />
- <s:iterator value="#list" id="n">
- <s:property value="n"/>
- </s:iterator>

- 生成一个Map对象：
- <s:set name="foobar" value="#{'foo1':'bar1', 'foo2':'bar2'}" />
- <s:iterator value="#foobar" >
- <s:property value="key"/>=<s:property value="value"/>
- </s:iterator>

- Set标签用于将某个值放入指定范围。
- scope：指定变量被放置的范围，该属性可以接受application、session、request、page或action。如果没有设置该属性，则默认放置在OGNL Context中。
- value：赋给变量的值.如果没有设置该属性,则将ValueStack栈顶的值赋给变量。

采用OGNL表达式判断对象是否存在于集合中

- 对于集合类型，OGNL表达式可以使用in和not in两个元素符号。其中，in表达式用来判断某个元素是否在指定的集合对象中；not in判断某个元素是否不在指定的集合对象中，如下所示。
- **in表达式：**
- <s:if test="'foo' in {'foo','bar'}">
- 在
- </s:if>
- <s:else>
- 不在
- </s:else>

- **not in表达式：**
- <s:if test="'foo' not in {'foo','bar'}">
- 不在
- </s:if>
- <s:else>
- 在
- </s:else>

OGNL表达式的投影功能

- 除了in和not in之外，OGNL还允许使用某个规则获得集合对象的子集，常用的有以下3个相关操作符。
- ?：获得所有符合逻辑的元素。
- ^：获得符合逻辑的第一个元素。
- $：获得符合逻辑的最后一个元素。