

OGNL上下文

ContextMap

明确：它是struts2中存放数据的容器。我们的所有数据都存放在该容器中。

OGNL上下文对象的结构：它是一个Map结构。Map的key是String。Map的value是Object。

OGNL上下文对象：ContextMap。封装着我们一次请求会用到的所有数据。

ContextMap中封装的数据：

context map包含的数据：

- 1、application 对应的是应用域中的所有数据。 Map结构
- 2、session 对应的是会话域中的所有数据。 Map结构
- 3、request 对应的是请求域中的所有数据 Map结构
- 4、**value stack(root)** 它是struts2独有的。叫做值栈对象。它实现了栈的特性（先进后出） 非常重要
- 5、action (the current action) 它在值栈中已经存在。当前执行的动作类对象 这是一个对象。
- 6、parameters 由于有模型驱动的存在，所以不用它， 请求参数(request.getParamterMap())能获取到的数据） Map结构
- 7、attr (searches page, request, session, then application scopes)它存的是四个域的数据组合。全域搜索。由小到大。 Map结构由于上面已经包含了3大域数据，所以这个里面的三个域不用了。由于我们Action中，没有页面的事，所以page域也不用了。

明确：OGNL表达式它在获取数据时，就是在OGNL上下文中查找数据。

在OGNL上下文之中，有两种方式可以获取域的map。

一种是短key：是给开发人员用的。

一种是长key：是给框架自己用的。

ActionContext对象

明确：动作类是多例的。每个线程一个实例。

ActionContext它是struts2框架提供的一个工具类。

它能是开发人员快速的访问ContextMap。以及ContextMap中所包含的内容。

创建时间点：

核心控制器：StrutsPrepareAndExecuteFilter的doFilter方法执行时创建

由于动作类是多例的，如何实现每个线程获取到自己的对象

使用ThreadLocal，把它绑定到当前线程上。

如何获取ActionContext：使用该类中的静态方法getContext(),从当前线程上获取该对象。

ContextMap的map部分的操作

* 包括:

ContextMap的存取值

在动作方法中存入数据

```
public String demo1(){
```

需求: 往ContextMap中存入数据, 存入 key: contextMap, value: hello context map

1. 获取ActionContext对象

```
ActionContext context = ActionContext.getContext();
```

2. 使用该对象的put方法往map中存入数据

```
context.put("contextMap", "hello context map");
```

```
public class DemolAction extends ActionSupport {
```

```
/**
```

```
 * map部分的操作
```

```
 * 明确: 我们把ContextMap称之为大Map, 把Application, Session这些都称之为小map
```

```
 */
```

```
public String demo1() { 动作类方法
```

```
/*
```

```
 * 需求:
```

```
 * 往ContextMap中存入数据
```

```
 * 存入key: contextMap
```

```
 * value: hello context map
```

```
 */
```

```
//1. 获取ActionContext对象
```

```
ActionContext context = ActionContext.getContext();
```

```
//2. 使用该对象的put方法往map中存入数据
```

```
context.put("contextMap", "hello context map");
```

获取ActionContext对象, 该对象对
ContextMap (OGNL上下文) 进行了
封装

OGNL上下文对象可以直接操作四大域
对象 进行数据的共享存取操作

在页面**取contextMap**里的值:

OGNL表达式获取ContextMap数据: `<s:property value="#ContextMap"/>`

application域的存取值

* 需求: 往application这个小Map中存入数据

* 存入的key: applicationMap 存入的value: hello application

```
*/
```

//方式一: 使用Map

```
Map<String, Object> application = ac.getApplication();
```

```
application.put("applicationMap", "我是一个applicationMap里存的数据");
```

//方式二: 使用原始ServletAPI对象: ServletContext

```
ServletContext application2 = ServletActionContext.getServletContext();
```

```
application2.setAttribute("applicationAttr", "我是一个applicationMap使用  
原始ServletAPI对象存的数据");
```

在页面取application域里的值

获取key为applicationMap的值: `<s:property value="#application.applicationMap"/>`
`
`

获取key为applicationAttr的

值: `<s:property value="#application.applicationAttr"/>`

session域的存取值

* 需求: 往session小Map中存入数据

//方式一: 使用Map

```
Map<String,Object> sessionMap = ac.getSession();
```

```
sessionMap.put("sessionMap","我是一个sessionMap里存的数据");
```

//方式二: 使用使用原始ServletAPI对象: `ServletActionContext.getRequest()`

```
HttpSession session = ServletActionContext.getRequest().getSession();
```

```
session.setAttribute("sessionAttr","我是一个sessionAttr使用原始  
ServletAPI对象存的数据");
```

在页面取session域里的值

获取key为sessionMap的值: `<s:property value="#session.sessionMap"/>`

`
`

获取key为sessionAttr的值: `<s:property value="#session.sessionAttr"/>`

ValueStack值栈（List结构）

它是struts2中独有的一个对象，也是用于封装数据用的。它的结构是一个List。并且实现了栈的特点（先进后出）。

获取值栈的数据:

明确: 我们都是根据值栈中对象的属性名称获取属性的值。

借助struts2的s:property标签

注意:

取ValueStack中的数据，使用OGNL表达式时，不要写#号。

而是直接写属性名称，得到的就是属性的值。

它的查找方式是：

从栈顶逐个对象的属性开始查找，只要找到了第一个，返回数据。并且不再继续查找

在代码中获取ValueStack并往ValueStack里面压栈

1、创建一个对象

```
private User user = new User();
```

2、获取值栈对象

```
ApplicationContext ac = ApplicationContext.getContext();
```

```
ValueStack vs = ac.getValueStack();
```

3、执行压栈操作

```
vs.push(user);
```

在页面取出对象ValueStack的值

user的name是：<s:property value="name"/>

user的age是：<s:property value="age"/>

user的age是：<s:property value="hobby"/>

2、获取指定位置的属性

（栈里有两个不同对象的属性，属性名都相同，属性值不同时可以通过以下方法获取指定属性的值）

[x].属性名称。

x指的是索引。指的是List中对象的索引。

Student的name是：<s:property value="[0].name"/>

动作类中的name是：<s:property value="[1].name"/>

3、当我们使用s:property标签之后，没有指定value属性时获取的就是栈顶对象。

```
<s:property/>
```

4、当我们没有操作值栈时，默认的栈顶对象是：当前执行的动作类！

Struts ValueStack Debug

Value Stack Contents

Object	Property Name	Property Value
com.rong.User	birthday	null
	name	小李子
	age	35
	hobby	写代码
com.web.UserAction2	container	There is no read method for container
	errorMessages	[]
	actionErrors	[]
	actionMessages	[Error setting expression 'age' with value [''], Error setting expression 'birthday' with value [''], Error setting expression 'name' with value [''],]]
	fieldErrors	{}
	texts	null
	name	当前类的成员属性
	age	35
	birthday	null
	hobby	写代码

如果没有压栈那么就是当前动作类在栈顶，想要获取当前动作类的属性值那必须要在动作类中有相关属性的get方法

Struts ValueStack Debug

Value Stack Contents

Object	Property Name	Property Value
com.web.UserAction2	container	There is no read method for container
	errorMessages	[]
	actionErrors	[]
	actionMessages	[Error setting expression 'age' with value [''], Error setting expression 'birthday' with value [''], Error setting expression 'name' with value [''],]]
	fieldErrors	{}
	texts	null
	name	当前类的成员属性
	age	35
	birthday	null
	hobby	写代码

```
user.name是: <s:property value="user.name"/>
<br/>
user.name是: <s:property value="user.age"/>
<br/>
user.name是: <s:property value="user.hobby"/>
```

如果栈顶的属性是一个对象 则需要通过对象.属性的方法取值

```

/**
 * Struts2动作类
 */
public class UserAction2 extends ActionSupport{

    //一个人的对象（属性：姓名 小李子 年龄 35 爱好 写代码）
    private User user = new User("小李子",35,"写代码");
    private String name="当前类的成员属性";

    //动作类里的成功响应方法
    public String register1() {
        return SUCCESS;
    }

    //获取user对象的get方法
    public User getUser() {
        return user;
    }

    public String getName() {
        return name;
    }
}

```

没有手动压栈的动作类，在栈顶默认的对象就是当前动作类

必须提供的get方法，否则无法获取栈顶的属性

在类中获取的方法：

```

ActionContext ac = ActionContext.getContext();

```

```

ValueStack vs = ac.getValueStack();

```

在类中对应取值的方法

```

Object value = vs.findValue("name");

```

struts2对EL表达式的改变

改变前在域中查找数据的顺序：

page域——>request域——>session域——>application域

改变后在请求域找不到会先去值栈找找找不到去Map里找然后再去后面的域里找：

page域——>request域——>ValueStack——>ContextMap——>session域——>application域

注意：

它只搜索大map中有没有key相同的。

当我们使用OGNL表达式不写#号时，它会先去值栈中查找属性名称相同的。取属性的值。

如果找不到，还会把表达式看成是key，在ContextMap查找。

但是实际开发中，**建议找值栈就不写#号。找ContextMap就写#号**

获取Map中的数据（包括ContextMap以及他们的小Map,但是不包含ValueStack对象的）

明确：

OGNL表达式只能出现struts2的标签中

借助s:property标签获取map中的数据

它只能根据map中的key，获取value。

注意：

获取Map中的数据，需要使用#key。

取出来的是map中的value