OGNL 概述

1、之前 web 阶段，学习过 EL 表达式，EL 表达式在 jsp 中获取域对象里面的值

2、OGNL 是一种表达式，这个表达式功能更加强大

(1)、在 struts2 里面操作值栈数据

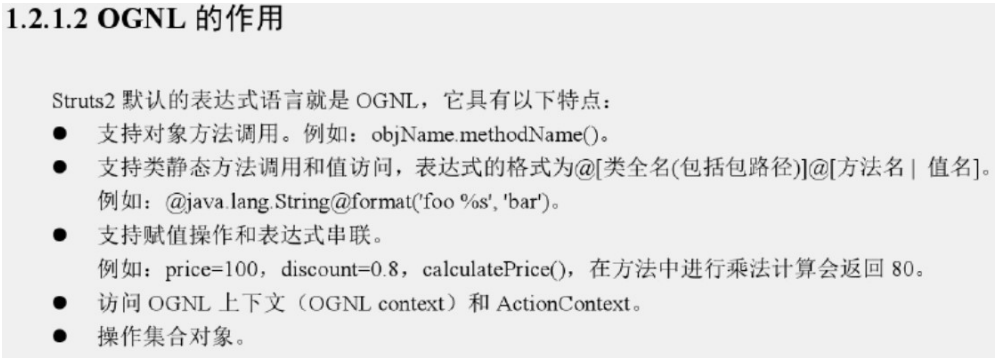
(2)、一般在 ognl 在 struts2 操作，和 struts2 标签一起使用操作值

3、OGNL 不是 struts2 的一部分，单独的项目，经常和 struts2 一起使用

(1)、使用 ognl 时候首先导入 jar 包，struts2 提供 jar 包

OGNL 入门案例

1、ognl 的作用



1.2.1.2 OGNL 的作用

Struts2 默认的表达式语言就是 OGNL，它具有以下特点：

- 支持对象方法调用。例如：objName.methodName()。
- 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')。
- 支持赋值操作和表达式串联。
  例如：price=100，discount=0.8，calculatePrice()，在方法中进行乘法计算会返回 80。
- 访问 OGNL 上下文（OGNL context）和 ActionContext。
- 操作集合对象。

2、使用 ognl+struts2 标签实现计算字符串长度

(1)、在 java 代码中，调用字符串 .length();

(2)、在 ognl 中，使用 objectName.methodName();

3、使用 struts2 标签

(1)、使用 jstl 时候，需要导入 jar 包之外，在 jsp 页面中引入标签库，同理，在使用 struts2 标签时候，在 jsp 中引入标签库

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

(2)、使用 struts2 标签实现操作

```
<!-- 使用ognl+struts2标签实现计算字符串长度
        value：ognl表达式
 -->
```

```
4       <s:property value="'hahaha'.length()"/>
```

什么是值栈

1、之前 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 的区别

(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）

获取值栈对象

1、获取值栈对象有多种方式

(1)、常用方式：使用 actionContext 类里面的方式得到值栈对象

```
1       ActionContext ctxt = ActionContext.getContext();
2       ValueStack valueStack = ctxt.getValueStack();
```

2、每个 Action 有且只有一个值栈对象

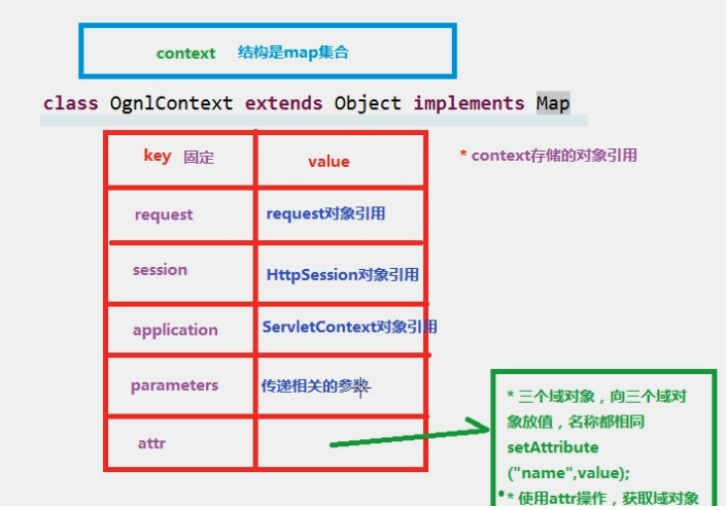
值栈内部结构

1、值栈分为两部分

第一部分 root 结构是 list 集合

(1)、一般操作是 root 里 面数据

第二部分 context， 结构 map 集合

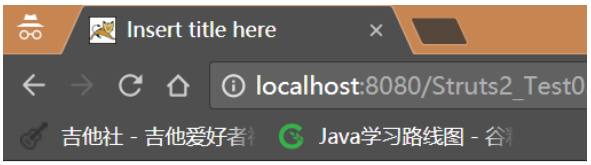


2、struts2 里面标签 s:debug， 使用这个标签可以查看值栈结构和存储值

(1)访问 Action, 执行 Action 的方法返回值，配置返回值到 jsp 页面中，在 jsp 页面中使用这个标签

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@taglib uri="/struts-tags" prefix="s" %>
http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
    <s:debug></s:debug>
</body>
</html>
```



[Debug]

点击链接即可查看 结构

向值栈中放数据

1、向值栈放数据的多种方式

第一种：获取值栈对象，调用值栈里面的 set 方法

```
        ActionContext context = ActionContext.getContext();
        ValueStack valueStack = context.getValueStack();
        valueStack.set("daiwei", "666");
```

第二种：获取值栈对象，调用值栈对象里面的 push 方法

```
        ActionContext context = ActionContext.getContext();
        ValueStack valueStack = context.getValueStack();
        valueStack.push("daiwei6666");
```

第三种：在 Action 定义变量，生成变量的 get 方法（建议使用）

```
    private String username;

    public String getUsername() {
        return username;
    }


    public void setUsername(String username) {
        this.username = username;
    }

    @Override
    public String execute() throws Exception {
//      ActionContext context = ActionContext.getContext();
//      ValueStack valueStack = context.getValueStack();
//      valueStack.set("daiwei", "666");
//      ActionContext context = ActionContext.getContext();
//      ValueStack valueStack = context.getValueStack();
//      valueStack.push("daiwei6666");
        this.username ="daiwei666";

        return Action.SUCCESS;
    }
```

2、向值栈中放对象

1、实现步骤

第一步：定义对象变量

第二步：生成变量的 get 方法

第三步：在执行方法向里面设置对象

```
public class ValueStake extends ActionSupport {

    private User user = new User();

    public User getUser() {
```

```
        return user;
    }

    @Override
    public String execute() throws Exception {
        user.setUsername("daiwei");
        user.setPassword("123456");
        user.setAge(21);
        return Action.SUCCESS;
    }
}
```

3、向值栈中放 LIst 集合

第一步：定义 list 集合变量

第二步： 生成变量的 get 方法

第三步：在执行的方法里面向 list 集合设置值

```

public class ValueStake extends ActionSupport {


    private List<User> users = new ArrayList<>();


    public List<User> getUsers() {
        return users;
    }

    public void setUsers(List<User> users) {
        this.users = users;
    }


    @Override
    public String execute() throws Exception {
        User user = new User();
        user.setUsername("xxxx");
        user.setPassword("xxxx");
        user.setAge(22);
```

```
        User user1 = new User();
        user1.setUsername("yyyy");
        user1.setPassword("yyyy");
        user1.setAge(22);
        User user2 = new User();
        user2.setUsername("zzzz");
        user2.setPassword("zzzz");
        user2.setAge(22);
        users.add(user);
        users.add(user1);
        users.add(user2);
        return Action.SUCCESS;
    }
}
```

从值栈中获取数据

1、使用 struts2 的标签 + ognl表达式获取值栈数据

<s:property value = "ognl表达式"/>

2、从值栈中获取字符串

1、 后端 java 代码部分

```
package daiwei.learning.struts2;

import com.opensymphony.xwork2.Action;

public class GetValueFromVS {

    private String username;

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getValue() {
```

```
        this.username = "daiwei66666";
        return Action.SUCCESS;
    }
}
```

2、jsp 中使用调用值栈标签：`<s:property value="username"/>`

3、从值栈获取对象

1、后端 java 代码

```
package daiwei.learning.struts2;

import com.opensymphony.xwork2.Action;

import daiwei.learning.pojo.User;

public class GetValueFromVS {

    private User user = new User();

    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }

    public String getValue() {

        user.setUsername("daiwei");
        user.setPassword("1231231");
        user.setAge(22);
        return Action.SUCCESS;
    }
}
```

2、jsp 中使用的标签

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

```
<s:property value="user.password"/>
<s:property value="user.age"/>
```

4、从值栈中获取 list 集合

1、后端 java 放集合代码

```
package daiwei.learning.struts2;

import java.util.ArrayList;
import java.util.List;

import com.opensymphony.xwork2.Action;

import daiwei.learning.pojo.User;

public class GetValueFromVS {

    private List<User> users = new ArrayList<User>();

    public List<User> getUsers() {
        return users;
    }

    public void setUsers(List<User> users) {
        this.users = users;
    }

    public String getValue() {
        User user = new User();
        user.setUsername("daiwei");
        user.setPassword("1231212");
        user.setAge(22);
        users.add(user);

        User user1 = new User();
        user1.setUsername("cjx");
        user1.setPassword("1s23fd31");
        user1.setAge(21);
        users.add(user1);

        User user2 = new User();
        user2.setUsername("qy");
```

```
        user2.setPassword("1dfdf12sdfds31");
        user2.setAge(24);
        users.add(user2);
        return Action.SUCCESS;
    }
}
```

2、jsp展示 List 集合

第一种方式：

```
    <s:property value="users[0].username"/>
    <s:property value="users[0].password"/>
    <s:property value="users[0].age"/>
    <br>
    <s:property value="users[1].username"/>
    <s:property value="users[1].password"/>
    <s:property value="users[1].age"/>
    <br>
    <s:property value="users[2].username"/>
    <s:property value="users[2].password"/>
    <s:property value="users[2].age"/>
```

第二种方式：

```
    <!-- 第二种方式通过使用类似 jstl 中 foreach 的 iterator迭代器实现  -->
    <s:iterator value="users">
        <s:property value="username"/>
        <s:property value="password"/>
        <s:property value="age"/>
        <br>
    </s:iterator>
```

第三种方式（常用）：

```
    <s:iterator value="users" var="user">
        <!--
            遍历值栈 list 集合，得到每个User对象
            机制：把每次遍历出来的user对象放到 context 里面
```

```
            获取context里面数据特点：写ognl表达式
            使用特殊符号：#
        -->
        <s:property value="#user.username"/>
        <s:property value="#user.password"/>
        <s:property value="#user.age"/>
        <br/>
    </s:iterator>
```

5、其他操作

1、使用 set 方法想值栈放数据，获取

```
valueStack.set("xxxx", "method");
```

```
<s:property value="xxxx"/>
```

2、使用 push 方法向值栈放数据，获取

```
valueStack.push("6666");
```

(1)、使用 push 方法设置值，没有名称，只有设置的值
(2)、向值栈放数据，把向值栈放数据存到数组里面，数组名称 top，根据数组获取值

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

EL 表达式可以获取值栈数据（为什么）

1、EL 表达式获取域对象值
    ${值的key}
2、向域对象里面放值使用 setAttribute 方法，获取值使用 getAttribute 方法
3、底层增强 request 对象里面的方法 getAttribute 方法
    1)、首先从 request 域获取值，如果获取到，直接返回

2)、如果从 request 域获取不到值，到值栈的值域取出来，把值放到域对象里面

ognl表达式 # 使用

1、向 Request 域放值

2、在页面中使用 ognl 获取（获取context里面数据特点：写ognl表达式，使用#）

```
<s:property value="#user.username"/>
<s:property value="#user.password"/>
<s:property value="#user.age"/>
```

ognl 表达式 % 使用

1、在 struts2 标签中表单标签

1）在 struts2 标签里面使用 ognl 表达式，如果直接在 struts2 表单标签里面使用 ognl 表达式不识别，只有 % 之后才会识别

```
<s:textfield name="username" value="%{#request.req}"/>
```