**Struts2的值栈（存数据） 掌握：**

1. 值栈是什么：

**值栈是struts2的数据中转站，栈中自动保存了当前Action对象和其他相关对象（包括常用的web对象的引用，如request，session，application等），也可以手动保存自己的数据对象，同时也可以随时随地的将对象从值栈中取出或操作（通过ognl表达式）**

**值栈（valuestack）：实际上是一个接口对象的称呼（接口是valueStack类，实现类是ognlValueStack类，该对象是struts2利用ognl的基础或者说struts2中ognl使用时基于值栈完成的）。（即在struts2中的ognl操作都是基于值栈完成的）**

1. **struts2框架将valueStack对象保存在request域中，键为“struts.valueStack”，即值栈是request域中的一个对象，一个请求对应一个Action实例和一个值栈对象。**

**值栈的两种获取方式：**

1. **request.getAttribute(“struts.valueStack”);用的较少**
2. **ActionContext.getContext().getValueStack();用的非常多（此为取得root栈，取得map栈要用Actioncontext.getContext()）。但是底层还是使用的第一种方式完成。（ActionContext:Action 上下文，工具类）**

范例：创建值栈

**package** wp.demo\_03;

**import** org.apache.struts2.ServletActionContext;

**import** com.opensymphony.xwork2.ActionContext;

**import** com.opensymphony.xwork2.ActionSupport;

**import** com.opensymphony.xwork2.util.ValueStack;

**public** **class** ValueStackAction\_01 **extends** ActionSupport{

@Override

**public** String execute() **throws** Exception {

//获取值栈

//1.从request域中获取值栈 先获取request

ValueStack vs = (ValueStack)**ServletActionContext.**

***getRequest*()**.getAttribute("struts.valueStack") ;

//2.通过ActionContext对象获取值栈

ValueStack vs2 = ActionContext.*getContext*().getValueStack() ;

System.***out***.println(vs == vs2);

System.***out***.println(vs.hashCode());

System.***out***.println(vs2.hashCode());

**return** ***NONE***;

}

}

范例：**1.root栈。使用push和set方式将数据压入栈中**

package wp.demo\_03;

import org.apache.struts2.ServletActionContext;

import com.opensymphony.xwork2.ActionContext;

import com.opensymphony.xwork2.ActionSupport;

import com.opensymphony.xwork2.util.ValueStack;

public class ValueStackAction\_02 extends ActionSupport{

@Override

public String execute() throws Exception {

**ValueStack vs = ActionContext.getContext().getValueStack() ;**

//往值栈中放数据

vs.push("文攀") ;

vs.push("小明") ;

vs.push("张三") ;

//使用K--V结构存放数据

vs.set("name", "文攀");

vs.set("age", 18);

return NONE;

}

}

范例：**使用map栈存放数据**

package wp.demo\_03;

import org.apache.struts2.ServletActionContext;

import com.opensymphony.xwork2.ActionContext;

import com.opensymphony.xwork2.ActionSupport;

import com.opensymphony.xwork2.util.ValueStack;

public class ValueStackAction\_03 extends ActionSupport{

@Override

public String execute() throws Exception {

//取得map栈对象

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

//把数据放在map栈中

context.put("name", "文攀");

context.put("age", 20);

return NONE;

}

}

**Struts2的值栈分为两种类型：**

1. **root栈 ：实际上是一个ArrayList结构，以栈的方式来存数据。（1.使用push方式将数据压入栈中 2.使用set方式把数据存在一个hashmap中，这个hashmap也压入栈中）**
2. **Map栈 ：实现map接口，以k-v的形式存放数据（1.使用put方式把数据放入map栈）**

**值栈的存值机制：**

对于OGNL表达式的操作都是基于OgnlContext(map栈)对象，访问的规则如下：

1. **如果访问root栈内容（CompoundRoot 对象栈内容），不需要#,直接通过元素名称来访问。**
2. **如果访问Map栈内容（如request，response，session，servletContext，parameters），需要使用#key来引用访问，例如：#request.name相当于request.getAttribute(“name”).**

**值栈的取值机制：**

范例：

**package** wp.demo\_04;

**import** org.apache.struts2.ServletActionContext;

**import** com.opensymphony.xwork2.ActionContext;

**import** com.opensymphony.xwork2.ActionSupport;

**import** com.opensymphony.xwork2.util.ValueStack;

**public** **class** ValueStackAction1 **extends** ActionSupport{

@Override

**public** String execute() **throws** Exception {

//取得值栈对象  **root栈**

**ValueStack vs1 = ServletActionContext.*getContext*().getValueStack() ;**

//向root栈中存入数据对象 按ArrayList方式存放

vs1.push("文攀");

vs1.push("张三");

//通过k---v的形式将数据放入root栈

vs1.set("name", "刘德华");

//向**map**栈放数据 用#的方式取出

**ActionContext.*getContext*().put("name", "李四");**

**return** ***SUCCESS***;

}

}

<?xml version=*"1.0"* encoding=*"UTF-8"* ?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<%@page import=*"wp.demo\_02.Custorm"*%>

<html xmlns=*"http://www.w3.org/1999/xhtml"*>

<head>

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

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

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"* />

<title>值栈的取值方式</title>

</head>

<body>

<h3>获取栈顶元素</h3>

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

<h3>获取栈中的下一个元素</h3>

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

<h3>取root栈中k---v形式存的数据</h3>

<s:property value = *"name"*/>

<h3>取map栈中的数据，需要用#</h3>

**<s:property value = *"#name"*/>**

</body>

</html>

在以上程序中通过push或则set（键值对方式）向root栈中存入对象数据，然后利用<s:property value = “[x].top/>通过[].top的形式取出。然后利用ActionContext.*getContext*().put("name", "李四");向map栈中加入数据元素。然后利用#的形式取出。

**注意：要取出root栈中的数据不需要加#，要取出map栈中的数据必须要加#.**

**值栈的默认收索：**

**在栈的数据收索当中：不加#的情况下，先收索root栈如果没有再收索map栈**

**加#的情况下，就直接收索map栈**

**实际开发中，单个数据直接放入root栈中，k--v形式的数据放入map栈中**

范例：

<?xml version=*"1.0"* encoding=*"UTF-8"* ?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<%@page import=*"wp.demo\_02.Custorm"*%>

<html xmlns=*"http://www.w3.org/1999/xhtml"*>

<head>

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

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

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"* />

<title>值栈的取值方式</title>

</head>

<body>

<h3>获取map栈中的数据</h3>

<h3>不加#的情况下，先收索root栈如果没有再搜索map栈</h3>

<h3>加#的情况下，就直接搜索map栈</h3>

<h3>实际开发中，单个数据直接放入root栈中，k--v形式的数据放入map栈中</h3>

<s:property value = *"name"*/>

</body>

</html>

ActionContext.getContext().put("name", "李四");

**在程序中只是向map值栈添加了数据，按照一般规定要取得map栈中的数据必须要加#。但是如果没有加#，就会先搜索root栈，如果root栈没有相关数据，则搜索map栈。**

**值栈的存取小结：**

值栈的作用就是数据的保存和获取（可以在任何地方获取）

注意点：怎么存就怎么取

1. **存进root栈，就不通过#获取<s:property value = “xxx”/>**
2. **存进map栈，就通过#获取<s:property value = “#xxx”/>**

如何向值栈保存数据

1. ValueStack.push(obj),保存数据到Root栈顶-压栈顶（对象本身）-匿名
2. ActionContext.getContext.put(key,value),保存数据到Map栈中
3. ValueStack.set(key,value):将数据保存到Root栈顶（数据对象自动封装为Map来保存，栈顶是个map，map里面有个属性是对象）---有名字。
4. ognl表达式如何获取值栈的数据：

**·jsp页面获取**

<s:property value = “name”/>先收索root栈对象属性（getter方法，getXxx-->xxx）.再收索map栈的key

1. <s:property value = “#name/>收索map的key
2. 通过[index].top指定访问root栈某层对象，例如：[0].top栈顶对象

·**Action代码获取（即java代码中获取值栈中的值）**

ValueStack.findValue[ognl表达式]:获取值.栈数据

获取root值栈中的值：Eg:String str = ActionContext.getContext.getValueStack.findString(“username”);

在java代码中获取map值栈中的值：

**String str = ActionContext.getContext.getValueStack.findString(“#username”);**

范例：

package wp.demo\_04;

import org.apache.struts2.ServletActionContext;

import com.opensymphony.xwork2.ActionContext;

import com.opensymphony.xwork2.ActionSupport;

import com.opensymphony.xwork2.util.ValueStack;

public class ValueStackAction2 extends ActionSupport{

@Override

public String execute() throws Exception {

//向map栈加入数据

**ActionContext.getContext().put("name", "李四");**

**ActionContext.getContext().put("username", "文攀");**

//root栈放数据 首先通过ServletActionContext.getContext.getValueStack()取得

//值栈,然后向值栈设置值

**ServletActionContext.getContext().getValueStack().set("username", "王五");**

**ActionContext.getContext().getValueStack().push(“赵六”);**

//从root栈中取数据

String str1 = ActionContext.getContext().getValueStack().findString("username") ;

//从map栈中取数据

String str2 = ActionContext.getContext().getValueStack().findString("#username") ;

System.out.println(str1);

System.out.println(str2);

return SUCCESS;

}

}

在以上程序中，向root值栈中存放数据是用**ServletActionContext.getContext().put(key,value)**。**或则ActionContext.getContext().getValueStack().set(key,value)或ActionContext.getContext().getValueStack().push(obj)**向map值栈中存放数据使用的**ActionContext.getContext().put(key,value);**取数据使用的都是 **ActionContext.getContext().getValueStack().findString("#username") ;**

**模型驱动优先于属性驱动：即当使用模型驱动自动为对象的属性赋值时，优先为对象的属性进行赋值，然后不会对驱动类中的同名属性进行赋值。**

范例：比较模型驱动对对象属性赋值和属性驱动的优先性：

<?xml version=*"1.0"* encoding=*"UTF-8"* ?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<%@page import=*"wp.demo\_02.Custorm"*%>

<html xmlns=*"http://www.w3.org/1999/xhtml"*>

<head>

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

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

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"* />

<title>模型驱动和属性驱动比较</title>

</head>

<body>

<h3>s:debug标签可以在HTML页面上查看值栈内容</h3>

<s:debug></s:debug>

</body>

</html>

**package** wp.demo\_05;

**import** com.opensymphony.xwork2.ActionSupport;

**import** com.opensymphony.xwork2.ModelDriven;

**public** **class** TestDrivern **extends** ActionSupport **implements** ModelDriven<User>{

/\*\*

\* 模型驱动优先于属性驱动接收参数

\* \*/

//模型驱动

User user = **new** User() ; //使用模型驱动必须手动创建对象

//属性驱动

String username ;

**public** User getUser() {

**return** user;

}

**public** **void** setUser(User user) {

**this**.user = user;

}

**public** String getUsername() {

**return** username;

}

**public** **void** setUsername(String username) {

**this**.username = username;

}

@Override

**public** User getModel() {

**return** user;

}

@Override

**public** String execute() **throws** Exception {

**return** ***SUCCESS***;

}

}

**通过以上比较可以知道，首先是对模型驱动User对象中的username属性赋值，对TestDrivern类中的属性username不赋值。所以模型驱动优先于属性驱动接收参数。**

**值栈的生命周期：**

·贯穿整个Action的生命周期，每一个Action类的对象实例都拥有一个ValueStack对象。

·Struts2框架将ValueStack对象保存在名为“struts.valueStack”的（request）请求属性中，即值栈是request中的一个对象，一个请求对应一个Action实例和一个值栈对象。

当有请求到达的时候，struts2会为每一个请求创建一个新的值栈，也就是说，值栈和请求时一一对应的。不同的请求，值栈也不一样，而值栈也封装了一次请求中所需要操作的所有数据。

**因此：值栈的尚明周期也就是request的生命周期，也就是Action的生命周期。**

范例：通过重定向跳转页面时，目标页面无法获取到Action中放入值栈的值。

@Override

**public** String execute() **throws** Exception {

//向root值栈中放入值

ActionContext.*getContext*().getValueStack().set("name","文攀");

**return** ***SUCCESS***;

}

<?xml version=*"1.0"* encoding=*"UTF-8"* ?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns=*"http://www.w3.org/1999/xhtml"*>

<head>

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

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

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"* />

<title>valuestack</title>

</head>

<body>

<!-- 从值栈中取出数据 -->

<s:property value = *"name"*/>

</body>

</html>

从以上程序可以知道，在action中将数据放入了root值栈，在运行了action以后，在jsp页面中将值栈中的数据取出。如果是进行了服务器跳转则可以从值栈中取出数据。若是进行了客户端跳转以后，值栈的生命周期结束则无法从值栈中取出数据。

**值栈的生命周期与request对象一致。**

**Struts2支持的表达式语言：**

Struts2支持一下几种表达式语言：

·OGNL：可以方便的操作对象属性的开源表达式语言。

·EL：可以方便的操作jsp页面四个域范围数据（request，page，session，application）

·JSTL：JSP 2.0集成的标准表达式语言。

Groovy:基于java平台的动态语言，他具有时下比较流行的动态语言（如Python，Ruby和smalltack等）的一些特性。

Struts2默认的表达式语言是OGNL，原因是他相对于其他表达式语言更加简单，强大，最重要的是可以直接操作值栈。

Struts2通过值栈存放数据，那如何去数据呢？通过ognl表达式来访问值栈中的数据：

在struts2中OGNL表达式三种特殊符号的使用方式：

**·#用法**

**·%用法**

**·$用法**

1. #用法

范例：

**package** wp.ognldemo\_01;

**import** org.apache.struts2.ServletActionContext;

**import** com.opensymphony.xwork2.ActionSupport;

**public** **class** OgnlAction1 **extends** ActionSupport{

@Override

**public** String execute() **throws** Exception {

//把数据放入request，session，application3中属性范围 map值栈

ServletActionContext.*getRequest*().setAttribute("name", "request数据");

ServletActionContext.*getRequest*().getSession().setAttribute("name", "session数据");

ServletActionContext.*getServletContext*().setAttribute("name", "application数据");

**return** ***SUCCESS***;

}

}

<?xml version=*"1.0"* encoding=*"UTF-8"* ?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns=*"http://www.w3.org/1999/xhtml"*>

<head>

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

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

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"* />

<title>ognl</title>

</head>

<body>

<h3>通过OGNL表达式或EL表达式获取request域中的数据</h3>

<s:property value = *"****#request.name****"*/> | **${requestScope.name}**

<h3>通过OGNL表达式或EL表达式获取session域中的数据</h3>

<s:property value = *"#session.name"*/> | ${session.name}

<h3>通过OGNL表达式或EL表达式获取application域中的数据</h3>

<s:property value = *"#application.name"*/> | ${application.name}

<h3>如何获取page域的数据，只能保存在当前页面</h3>

<%

**pageContext.setAttribute("name","page数据") ;**

%>

<h3>取出page属性范围的数据</h3>

**<s:property value = *"#attr.name"*/> | ${name}**

<h3>获取地址栏传来的信息</h3>

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

</body>

</html>

在以上程序中，在action中首先通过三种属性范围进行往map值栈中加入数据。然后在jsp页面利用ognl表达式+标签进行取出。比如要取出request属性范围内的name属性对应的值，就直接用#request.name则可以取出。或直接用EL表达式${requestScope.name}。但是在page属性范围中，只能保存在当前页面。所以只能在jsp页面上用JavaScript代码进行属性的保存，然后通过**#attr.name**的形式取出。

**获取地址栏的信息：**

利用<s:property value = *"****#parameters.name****"*/> 标签中的#parameters.name来获取地址栏传来的参数。

1. **%用法**（主要作用：强制解析）：

范例：利用%强制解析值栈中的内容

<?xml version=*"1.0"* encoding=*"UTF-8"* ?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns=*"http://www.w3.org/1999/xhtml"*>

<head>

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

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

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"* />

<title>ognl</title>

</head>

<body>

<h2>文本框获取值栈中的值,使用%强制解析值栈中的值</h2>

**<s:textfield value = *"%{username}"*></s:textfield>**

<h2>强制不解析</h2>

**<s:property value = *"%{'username'}"*/>**

</body>

</html>

@Override

**public** String execute() **throws** Exception {

ServletActionContext.*getContext*().getValueStack().set("username", "文攀");

**return** ***SUCCESS***;

}

在以上程序中使用了***"%{username}"表示强制性的将username解析为值栈中的username，"%{'username'}"表示不强制解析为值栈中的内容，即为原来设置的内容***

1. **OGNL对值栈的操作三$的使用：**

·此处的$不能理解为EL表达式，因为此处的$必须使用在struts特有的文件中，只能用在struts2的文件中。

$主要作用：允许我们在配置文件中使用OGNL表达式，换句话说$可以在xml文件中获取值栈的值。

配置文件主要是指：struts.xml 国际化文件(xxx.properties) 校验配置文件xxx-validation.xml

范例：在资源文件中使用：在页面读取携带参数-----国际化文件中使用

package wp.ognldemo\_01;

import org.apache.struts2.ServletActionContext;

import com.opensymphony.xwork2.ActionSupport;

public class OgnlAction3 extends ActionSupport{

@Override

public String execute() throws Exception {

//如果要在xml或则properties文件中获取值栈的值，值不能是中文

**ServletActionContext.getContext().getValueStack().set("productName", "iphone");**

return SUCCESS;

}

}

<?xml version=*"1.0"* encoding=*"UTF-8"* ?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns=*"http://www.w3.org/1999/xhtml"*>

<head>

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

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

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"* />

<title>ognl</title>

</head>

<body>

**<h3>获取国际化文件中的值</h3>**

**<s:text name=*"productInfo"*></s:text>**

</body>

</html>

**Message.properties文件：**

productInfo=\u6B22\u8FCE\u4F7F\u7528**${productName}**\u4EA7\u54C1

配置文件：

<!-- 配置国际化常量 -->

<constant name = *"struts.custom.i18n.resources"* value = *"message"*></constant>

<action name=*"ognl3"* class = *"wp.ognldemo\_01.OgnlAction3"*>

<result name = *"success"*>/a\_ognl/ognl\_3.jsp</result>

</action>

在以上程序中，先是使用了action向root值栈中保存了key为productName值为iPhone的值。然后在src下的Message.properties文件：中通过**${}**读取了值栈中名为productName的值，然后在jsp页面通过<s:text/>标签加上ognl表达式读取了**Message.properties文件：**

的值。然后通过配置struts.xml配置文件即可访问。

范例：在struts.xml文件中使用$

Action同上。

Jsp页面的获取方式：

<h3>获取url中的参数</h3>

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

配置文件：

<action name=*"ognl4"* class = *"wp.ognldemo\_01.OgnlAction4"*>

<!--获取值栈中的产品名称，当做参数传给页面 -->

<result name = *"success"* type = *"redirect"***>/a\_ognl/ognl\_4.jsp?productName=${productName}**</result>

</action>

在以上程序中，首先通过action向值栈中设置了值，然后通过类似地址栏的方式传入要获取的参数（即：**/a\_ognl/ognl\_4.jsp?productName=${productName}**）。然后通过**<s:property value = *"#parameters.productName"*/>标签和ognl表达式进行访问。**

1. **OGNL对struts2的值栈的操作方式四（利用EL表达式操作值栈）：**

范例：利用EL表达式操作值栈

package wp.ognldemo\_01;

import org.apache.struts2.ServletActionContext;

import com.opensymphony.xwork2.ActionSupport;

public class OgnlAction5 extends ActionSupport{

@Override

public String execute() throws Exception {

//如果要在xml或则properties文件中获取值栈的值，值不能是中文

ServletActionContext.getContext().getValueStack().set("productName", "OPPO");

return SUCCESS;

}

}

<body>

<h3>利用EL表达式操作值栈</h3>

**${productName}**

</body>

在以上程序中直接通过EL表达式进行了值栈的取值操作。

**原理说明：struts2对request进行了包装，对request的getAttribute方法进行了增强。Struts2框架提供StrutsRequestWrapper包装类。**