1.2 OGNL

1.2.1 OGNL 概述

1.2.1.1 什么是OGNL

对象导航图语言（**Object Graph Navigation Language），简称**OGNL**，是应用于Java中的一个开源的表达式语言（Expression Language），它被集成在Struts2等框架中，作用是对数据进行访问，它拥有类型转换、访问对象方法、操作集合对象等功能。

- OGNL: 对象图导航语言,比 EL 表达式强大很多倍的语言.
  - EL: 从域对象中读取数据 ${name} ${pageContext.request.contextPath}
  - OGNL: 调用对象的方法,获取 Struts2 值栈的数据.OGNL是第三方的表达式语言.

1.2.1.2 为什么学习 OGNL

OGNL（Object-Graph Navigation Language），可以方便地操作对象属性的开源表达式语言，使页面更简洁；

支持运算符（如+-*/），比普通的标志具有更高的自由度和更强的功能；

Struts 2默认的表达式语言是OGNL，原因是它相对其它表达式语言具有下面几大优势：

- 支持对象方法调用，如xxx.doSomeSpecial()；
- 支持类静态的方法调用和值访问，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')或@tutorial.MyConstant@APP_NAME；
- 支持赋值操作和表达式串联，如price=100, discount=0.8, calculatePrice(price*discount)，这个表达式会返回80；
- 访问OGNL上下文（OGNL context）和ActionContext；
- 操作集合对象。
- 可以直接new一个对象

1.2.1.3 OGNL使用要素

1. 表达式

   表达式是整个OGNL的核心，所有的OGNL操作都是针对表达式的解析后进行的。表达式会规定此次OGNL操作到底要**"干什么"**。

2. 根对象

   根对象可以理解为OGNL的操作对象。在表达式规定了“干什么”以后，你还需要指定到底**"对谁干"**

3. Context对象

   有了表达式和根对象，我们实际上已经可以使用OGNL的基本功能。例如，根据表达式对根对象进行取值或者设值工作。

   不过实际上，在OGNL的内部，所有的操作都会在一个特定的环境中运行，这个环境就是OGNL的上下文环境（Context）。说得再明白一些，就是这个上下文环境（Context），将规定OGNL的操作**“在哪里干”**。

1.2.2 OGNL 的 Java 环境入门(了解)

1.2.2.1 访问对象的方法

```
/**
* @Title: OgnlDemo1.java
* @Package com.admiral.struts2.ognl
* @Description:
* @author 白世鑫
* @date 2020-9-28
* @version V1.0
*/
package com.admiral.struts2.ognl;

import org.junit.Test;

import ognl.Ognl;
import ognl.OgnlContext;
import ognl.OgnlException;

public class OgnlDemo1 {

    @Test
    /**
     * OGNL 调用对象的方法
     */
    public void demo1() throws OgnlException {
        // 获取 Cotext
        OgnlContext context = new OgnlContext();
        // 获取 根对象
        Object root = context.getRoot();

        // 执行表达式
        Object object = Ognl.getValue("'HelloWorld'.length()", context, root);
        System.out.println(object);

    }
}
```

1.2.2.2 访问对象的静态方法

```
    @Test
    /**
     * OGNL 访问对象的静态方法
     */
    public void demo2() throws OgnlException {
        // 获取 Cotext
        OgnlContext context = new OgnlContext();
        // 获取 根对象
        Object root = context.getRoot();
```

```
        // 执行表达式:@类名@方法名
        Object object = Ognl.getValue("@java.lang.Math@random()", context,
root);
        System.out.println(object);

    }
```

1.2.2.3 访问 root 中的数据

```
/**
* @Title: User.java
* @Package com.admiral.domain
* @Description:
* @author 白世鑫
* @date 2020-9-29
* @version V1.0
*/
package com.admiral.domain;

public class User {

    private String username;
    private String password;

    public User() {
        super();
        // TODO Auto-generated constructor stub
    }

    public User(String username, String password) {
        super();
        this.username = username;
        this.password = password;
    }

    public String getUsername() {
        return username;
    }

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

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    @Override
    public String toString() {
        return "User [username=" + username + ", password=" + password + "]";
    }
```

```
}
```

```
    @Test
    /**
     * OGNL 访问root中的
     */
    public void demo3() throws OgnlException {
        // 获取 Cotext
        OgnlContext context = new OgnlContext();

        //执行表达式
        User user = new User("admin","123");
        context.setRoot(user);

        // 获取 根对象
        Object root = context.getRoot();

        Object username = Ognl.getValue("username", context, root);
        Object passowrd = Ognl.getValue("password", context,root);
        System.out.println(username + "    " + passowrd);

    }
```

1.2.2.4 访问 Context 中的数据

```
    @Test
    /**
     * OGNL 访问 context 中的数据
     */
    public void demo4() throws OgnlException {
        // 获取 Cotext
        OgnlContext context = new OgnlContext();
        // 获取 根对象
        Object root = context.getRoot();

        context.put("name", "小花花");

        Object obj = Ognl.getValue("#name", context, root);
        System.out.println(obj);

    }
```

1.2.3 OGNL的 Struts2 环境入门

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib uri="/struts-tags" prefix="s"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <h1>OGNL Struts2 环境入门</h1>
    <h3>访问方法</h3>
    <s:property value="'struts'.length()"/>
    <h3>访问静态方法</h3>
    <!-- Struts2中静态方法访问是关闭的,需要开启一个常量 -->
    <s:property value="@java.lang.Math@random()"/>
</body>
</html>
```

开启常量

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.3//EN"
    "http://struts.apache.org/dtds/struts-2.3.dtd">

<struts>
    <constant name="struts.ognl.allowStaticMethodAccess" value="true">
</constant>
</struts>
```

1.3 值栈

1.3.1 值栈的概述

1.3.1.1 什么是值栈

Struts2将XWork对Ognl的扩展这一套机制封装起来，这个对象叫ValueStack。

ValueStack实际上就是一个容器。它由Struts框架创建，当前端页面如jsp发送一个请求时，Struts的默认拦截器会将请求中的数据进行封装，并入ValueStack的栈顶。

Struts2在启动时，会创建一个ValueStack对象 当用户发送请求到对应的Action时，Struts2会把当前被请求的Action01放入CompoundRoot 对象的“栈空间”栈顶，请求结束，Action01会被清除。 （当下一次另一个请求到来时，Struts2会把该请求对应的Action02放入“栈顶”） 所以，我们可以通过Ognl表达式访问CompoundRoot对象栈顶的Action。

Struts2在请求到来时，首先会创建一个ValueStack； 然后，把当前的Action对象放入栈顶（CompoundRoot）； Struts2会把ValueStack存放在request中，属性为”struts.valueStack“， 所以，标记库可以访问到ValueStack Struts2的很多标记就是通过访问ValueStack获得数据的：

- ValueSack 其实类似于一个数据中转站(Struts2框架中的数据就保存到了 ValueStack 中)
  - ValueStack 是一个接口,实现类 OgnlValueStack
  - ValueStack 贯穿整个 Action 的生命周期(Action 一旦创建,就会创建一个 ValueStack 对象)

1.3.1.2 值栈的内部结构

- ValueStack 中有两个主要的区域:
  - root 区域　　:其实就是一个ArrayList.　里面一般放置对象.获取 root 数据时不需要加 #
  - context 区域　　:其实就是一个Map.　里面一般放置 web 开发常用的对象的引用.获取 context 区域数据时需要加 #
    - request
    - session
    - application
    - parameters
    - attr
    - root
- 所谓的操作 ValueStack 指的是操作 root 区域.

1.3.1.3 值栈与 ActionContext 的关系

- ServletContext　:Servlet的上下文
- ActionContext　:Action的上下文
  - 当请求过来的时候,执行过滤器中的 doFilter 方法,在这个方法中创建了 ActionContext 对象,在创建 ActionContext 的过程中,创建了 ValueStack 对象,将 ValueStack 对象传递给了 ActionContext 对象.所以,可以通过 ActionContext 获取 ValueStack.
  - ActionContext 之所以能够访问 Servlet 的 API (访问域对象中的数据),是因为其内部有 ValueStack 的引用.

1.3.1.4 获得值栈

- 通过 ActionContext 对象获取值栈

```
/**
* @Title: ValueStackDemo2.java
* @Package com.admiral.struts2.valuestack
* @Description: 获取值栈
* @author 白世鑫
* @date 2020-9-29
* @version V1.0
*/
package com.admiral.struts2.valuestack;

import org.apache.struts2.ServletActionContext;

import com.opensymphony.xwork2.ActionContext;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.util.ValueStack;
```

```
public class ValueStackDemo2 extends ActionSupport{

    @Override
    public String execute() throws Exception {
        //方式一:通过 ActionContext 获取值栈
        ValueStack valueStack1 = ActionContext.getContext().getValueStack();
        return NONE;
    }

}
```

- 通过 request 获取值栈

```
/**
* @Title: ValueStackDemo2.java
* @Package com.admiral.struts2.valuestack
* @Description: 获取值栈
* @author 白世鑫
* @date 2020-9-29
* @version V1.0
*/
package com.admiral.struts2.valuestack;

import org.apache.struts2.ServletActionContext;

import com.opensymphony.xwork2.ActionContext;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.util.ValueStack;

public class ValueStackDemo2 extends ActionSupport{

    @Override
    public String execute() throws Exception {
        //方式二:通过 requests 获取值栈
//      ValueStack valueStack2 = (ValueStack)
ServletActionContext.getRequest().getAttribute("struts.valueStack");
        ValueStack valueStack2 = (ValueStack)
ServletActionContext.getRequest().getAttribute(ServletActionContext.STRUTS_VALUE
STACK_KEY);
        return NONE;
    }
}
```

1.3.1.5 操作值栈--向值栈中存入数据

- 在 Action 中提供属性的 get 方法
    - 默认情况下,会将Action压入值栈.

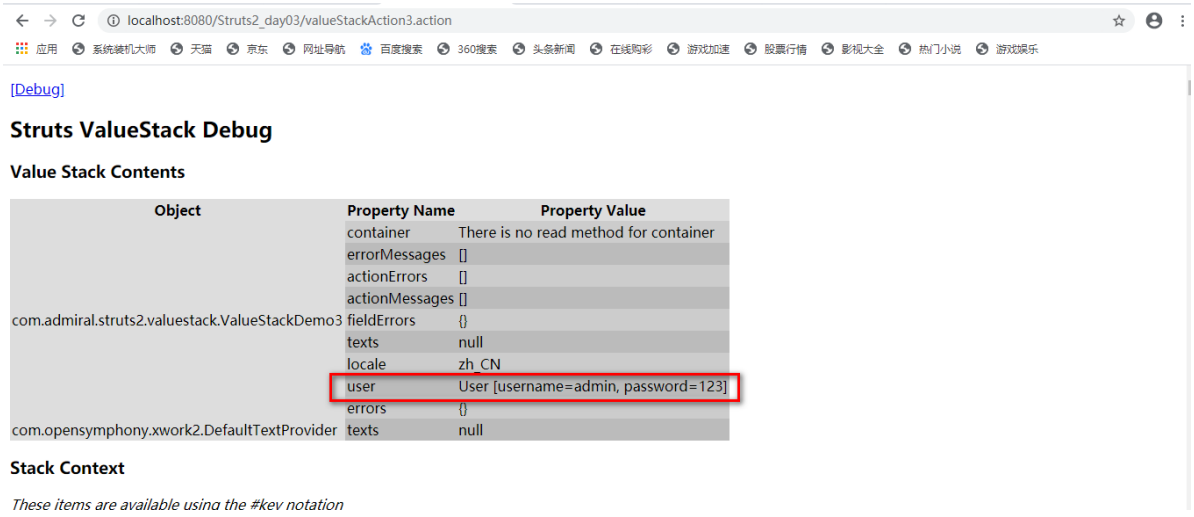
编写 Action

```java
/**
* @Title: ValueStackDemo3.java
* @Package com.admiral.struts2.valuestack
* @Description:
* @author 白世鑫
* @date 2020-9-29
* @version V1.0
*/
package com.admiral.struts2.valuestack;

import com.admiral.domain.User;
import com.opensymphony.xwork2.ActionSupport;

public class ValueStackDemo3 extends ActionSupport {

    private User user;

    public User getUser() {
        return user;
    }

    @Override
    public String execute() throws Exception {
        user = new User("admin", "123");
        return SUCCESS;
    }
}
```

编写配置文件

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.3//EN"
    "http://struts.apache.org/dtds/struts-2.3.dtd">

<struts>
    <package name="Demo1" extends="struts-default" namespace="/">
        <action name="valueStackAction1"
class="com.admiral.struts2.valuestack.ValueStackDemo1">
            <result>/demo1/success.jsp</result>
        </action>
        <action name="valueStackAction3"
class="com.admiral.struts2.valuestack.ValueStackDemo3">
            <result>/demo1/success.jsp</result>
        </action>
    </package>
</struts>
```

查看jsp



在页面获取值栈中的数据

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib uri="/struts-tags" prefix="s"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <s:debug></s:debug>
    <s:property value="user.username"/>
    <s:property value="user.password"/>
</body>
</html>
```

- 使用 ValueStack 本身的方法

编写 Action

```
/**
* @Title: ValueStackDemo4.java
* @Package com.admiral.struts2.valuestack
* @Description:
* @author 白世鑫
* @date 2020-9-29
```

```
* @version V1.0
*/
package com.admiral.struts2.valuestack;

import com.admiral.domain.User;
import com.opensymphony.xwork2.ActionContext;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.util.ValueStack;

public class ValueStackDemo4 extends ActionSupport {

    @Override
    public String execute() throws Exception {

        //向值栈中保存数据
        //使用 ValueStack 的方法
        ValueStack valueStack = ActionContext.getContext().getValueStack();

        User user = new User("xiaobai","111");

        //压栈,现在 user 对象在栈顶
        valueStack.push(user);

        //创建一个 map 集合,将集合压入栈顶
        valueStack.set("name", "xiaohuahua");

        return super.execute();
    }
}
```

编写配置文件

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.3//EN"
    "http://struts.apache.org/dtds/struts-2.3.dtd">

<struts>
    <package name="Demo1" extends="struts-default" namespace="/">
        <action name="valueStackAction1"
class="com.admiral.struts2.valuestack.ValueStackDemo1">
            <result>/demo1/success.jsp</result>
        </action>
        <action name="valueStackAction3"
class="com.admiral.struts2.valuestack.ValueStackDemo3">
            <result>/demo1/success.jsp</result>
        </action>
        <action name="valueStackAction4"
class="com.admiral.struts2.valuestack.ValueStackDemo4">
            <result>/demo1/success.jsp</result>
        </action>
    </package>
</struts>
```

修改JSP

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib uri="/struts-tags" prefix="s"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <s:debug></s:debug>
    <s:property value="username"/>
    <s:property value="password"/>
    <s:property value="name"/>
</body>
</html>
```

[Debug]

Struts ValueStack Debug

Value Stack Contents

Object	Property Name	Property Value
java.util.HashMap	empty	null
com.admiral.domain.User	password	111
	username	xiaobai
com.admiral.struts2.valuestack.ValueStackDemo4	container	There is no read method for container
	errorMessages	[]
	actionErrors	[]
	actionMessages	[]
	fieldErrors	{}
	texts	null
	locale	zh_CN
	errors	{}
com.opensymphony.xwork2.DefaultTextProvider	texts	null

Stack Context

1.3.1.6 获取值栈中的数据

- 获取值栈中的数据,在页面直接使用 OGNL 表达式即可
    - 获取 root 中的数据

编写 Action

```
/**
* @Title: ValueStackDemo5.java
* @Package com.admiral.struts2.valuestack
* @Description:
* @author 白世鑫
* @date 2020-9-29
* @version V1.0
*/
package com.admiral.struts2.valuestack;
```

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

import com.admiral.domain.User;
import com.opensymphony.xwork2.ActionContext;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.util.ValueStack;

public class ValueStackDemo5 extends ActionSupport {

    @Override
    public String execute() throws Exception {

        User user = new User("admin", "xiaobaibai");
        ValueStack valueStack = ActionContext.getContext().getValueStack();
        valueStack.push(user);

        //保存集合
        List<User> users = new ArrayList<User>();
        users.add(new User("aaa", "111"));
        users.add(new User("bbb", "222"));
        users.add(new User("ccc", "333"));
        valueStack.set("users", users);

        return super.execute();
    }
}
```

编写配置文件

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.3//EN"
    "http://struts.apache.org/dtds/struts-2.3.dtd">

<struts>
    <package name="Demo1" extends="struts-default" namespace="/">
        <action name="valueStackAction1"
class="com.admiral.struts2.valuestack.ValueStackDemo1">
            <result>/demo1/success.jsp</result>
        </action>
        <action name="valueStackAction3"
class="com.admiral.struts2.valuestack.ValueStackDemo3">
            <result>/demo1/success.jsp</result>
        </action>
        <action name="valueStackAction4"
class="com.admiral.struts2.valuestack.ValueStackDemo4">
            <result>/demo1/success.jsp</result>
        </action>
        <action name="valueStackAction5"
class="com.admiral.struts2.valuestack.ValueStackDemo5">
            <result>/demo1/success2.jsp</result>
        </action>
    </package>
</struts>
```

编写 JSP 页面

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib uri="/struts-tags" prefix="s"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <s:debug></s:debug>
    <s:property value="username"/>
    <s:property value="password"/><br/>
    <s:property value="users[0].username"/>
    <s:property value="users[0].password"/><br/>
    <s:property value="users[1].username"/>
    <s:property value="users[1].password"/><br/>
    <s:property value="users[2].username"/>
    <s:property value="users[2].password"/><br/>
</body>
</html>
```

- 获取 context 中的数据

编写 Action

```
/**
* @Title: ValueStackDemo5.java
* @Package com.admiral.struts2.valuestack
* @Description:
* @author 白世鑫
* @date 2020-9-29
* @version V1.0
*/
package com.admiral.struts2.valuestack;

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

import org.apache.struts2.ServletActionContext;

import com.admiral.domain.User;
import com.opensymphony.xwork2.ActionContext;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.util.ValueStack;
```

```
public class ValueStackDemo5 extends ActionSupport {

    @Override
    public String execute() throws Exception {

        User user = new User("admin", "xiaobaibai");
        ValueStack valueStack = ActionContext.getContext().getValueStack();
        valueStack.push(user);

        //保存集合
        List<User> users = new ArrayList<User>();
        users.add(new User("aaa", "111"));
        users.add(new User("bbb", "222"));
        users.add(new User("ccc", "333"));
        valueStack.set("users", users);


        //向 Context 中保存数据
        ServletActionContext.getRequest().setAttribute("name", "r小白");
        ServletActionContext.getRequest().getSession().setAttribute("name", "s小
红");
        ServletActionContext.getServletContext().setAttribute("name", "a小黄");

        return super.execute();
    }
}
```

编写 JSP

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib uri="/struts-tags" prefix="s"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <s:debug></s:debug>
    <s:property value="username"/>
    <s:property value="password"/><br/>
    <s:property value="users[0].username"/>
    <s:property value="users[0].password"/><br/>
    <s:property value="users[1].username"/>
    <s:property value="users[1].password"/><br/>
    <s:property value="users[2].username"/>
    <s:property value="users[2].password"/><br/>

    <!-- 获取 Context 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"/>
</body>
```

```
</html>
```

1.3.1.7 EL为何访问到值栈的数据

- 因为 Struts2 底层对request.getAttribute(String name); 方法进行了增强

```
 * $Id$

package org.apache.struts2.dispatcher;

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

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletRequestWrapper;

import static org.apache.commons.lang3.BooleanUtils.isTrue;

/**
 * <!-- START SNIPPET: javadoc -->
 *
 * All Struts requests are wrapped with this class, which provides simple JSTL accessibility. This is because JSTL
 * works with request attributes, so this class delegates to the value stack except for a few cases where required to
 * prevent infinite loops. Namely, we don't let any attribute name with "#" in it delegate out to the value stack, as it
 * could potentially cause an infinite loop. For example, an infinite loop would take place if you called:
 * request.getAttribute("#attr.foo").
 *
 * <!-- END SNIPPET: javadoc -->
 *
 */
public class StrutsRequestWrapper extends HttpServletRequestWrapper {

    private static final String REQUEST_WRAPPER_GET_ATTRIBUTE = "__requestWrapper.getAttribute";
    private final boolean disableRequestAttributeValueStackLookup;

    /**
     * The constructor
     * @param req The request
     */
    public StrutsRequestWrapper(HttpServletRequest req) {
        this(req, false);
    }

    /**
     * The constructor
     * @param req The request
     * @param disableRequestAttributeValueStackLookup flag for disabling request attribute value stack lookup (JSTL acces
     */
    public StrutsRequestWrapper(HttpServletRequest req, boolean disableRequestAttributeValueStackLookup) {
        super(req);
        this.disableRequestAttributeValueStackLookup = disableRequestAttributeValueStackLookup;
    }

    /**
     * Gets the object, looking in the value stack if not found
     *
     * @param key The attribute key
     */
    public Object getAttribute(String key) {
        if (key == null) {
            throw new NullPointerException("You must specify a key value");
        }

        if (disableRequestAttributeValueStackLookup || key.startsWith("javax.servlet")) {
            // don't bother with the standard javax.servlet attributes, we can short-circuit this
            // see WW-953 and the forums post linked in that issue for more info
            return super.getAttribute(key);
        }

        ActionContext ctx = ActionContext.getContext();
        Object attribute = super.getAttribute(key);

        if (ctx != null && attribute == null) {
            boolean alreadyIn = isTrue((Boolean) ctx.get(REQUEST_WRAPPER_GET_ATTRIBUTE));

            // note: we don't let # come through or else a request for
            // #attr.foo or #request.foo could cause an endless loop
            if (!alreadyIn && !key.contains("#")) {
                try {
                    // If not found, then try the ValueStack
                    ctx.put(REQUEST_WRAPPER_GET_ATTRIBUTE, Boolean.TRUE);
                    ValueStack stack = ctx.getValueStack();
                    if (stack != null) {
                        attribute = stack.findValue(key);
                    }
                } finally {
                    ctx.put(REQUEST_WRAPPER_GET_ATTRIBUTE, Boolean.FALSE);
                }
            }
        }
        return attribute;
    }
}
```

1.4 OGNL 中的特殊字符

1.4.1 #号

1.4.1.1 获取 Context 中的数据

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib uri="/struts-tags" prefix="s"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <h1>#号的用法 </h1>
    <h3>获取 Context 中的数据</h3>
    <%
        request.setAttribute("name", "小花花");
    %>
    <s:property value="#request.name"/>
</body>
</html>
```

1.4.1.2 构建 List 集合

```
    <h3>构建 List 集合</h3>
    <s:iterator var="i" value="{'aa','bb','cc'}">
        <s:property value="i"/>
    </s:iterator>
```

1.4.1.2 构建 Map 集合

```
    <h3>构建 Map 集合</h3>
    <s:iterator value="#{'11':'aa','22':'bb','33':'cc'}">
        <s:property value="key"/> -- <s:property value="value"/> <br/>
    </s:iterator>
```

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib uri="/struts-tags" prefix="s"%>
<!DOCTYPE html>
```

```
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <h1>#号的用法 </h1>
    <h3>获取 Context 中的数据</h3>
    <%
        request.setAttribute("name", "小花花");
    %>
    <s:property value="#request.name"/>
    <br/><br/>

    <h3>构建 List 集合</h3>
    <s:iterator var="i" value="{'aa','bb','cc'}">
        <s:property value="i"/> <br/>
    </s:iterator>

    <br/><br/>

    <h3>构建 Map 集合</h3>
    <s:iterator var="entry" value="#{'11':'aa','22':'bb','33':'cc'}">
        <s:property value="key"/> -- <s:property value="value"/> <br/>
        <s:property value="#entry.key"/> -- <s:property value="#entry.value"/>
<br/>
    </s:iterator>
    <hr/>
    性别:<input type="radio" name="sex1" value="男">男
    <input type="radio" name="sex1" value="女">女
    <hr/>
    <s:radio list="{'男','女'}" name="sex2" label="性别"></s:radio>
    <hr/>
    <s:radio list="#{'1':'男','2':'女'}" name="sex3" label="性别"></s:radio>

</body>
</html>
```

1.4.2 % 号

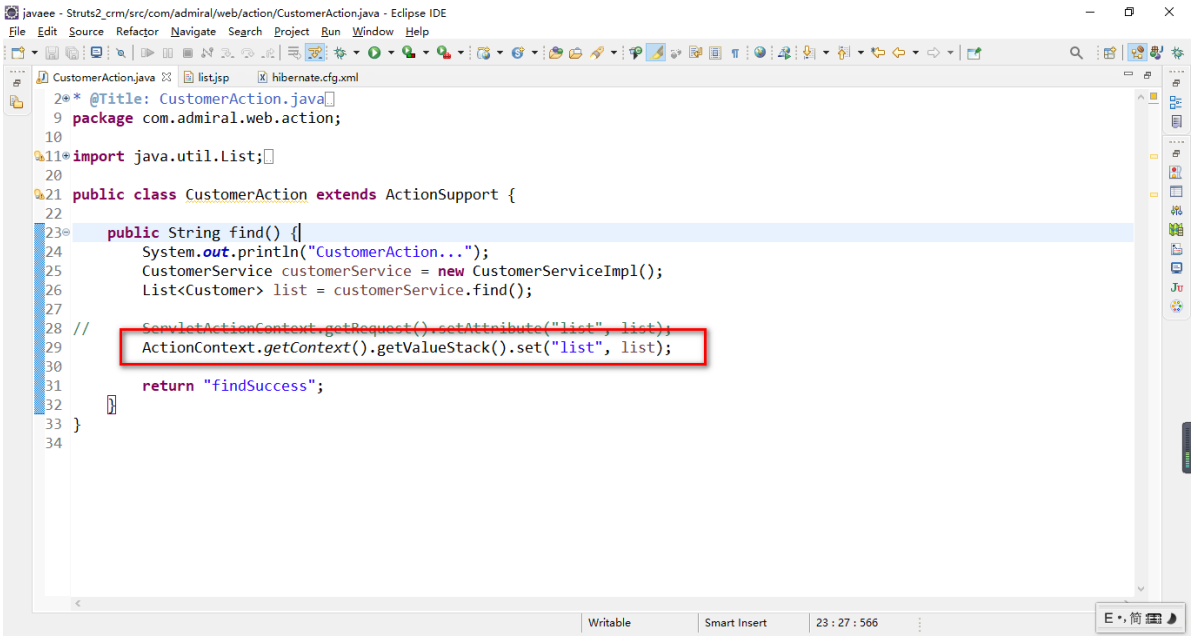
1.4.2.1 强制解析 OGNL

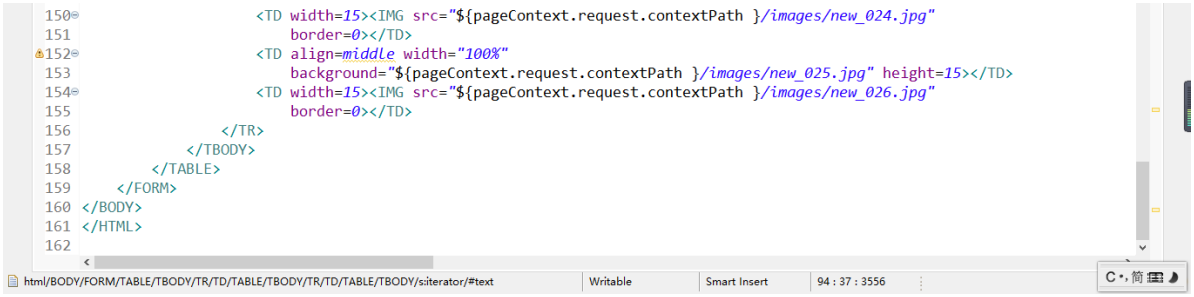
1.4.2.2 强制不解析 OGNL (没什么用)

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib uri="/struts-tags" prefix="s"%>
<!DOCTYPE html>
<html>
```

```
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <h1>%的用法</h1>
    <%
        request.setAttribute("username", "王小花");
    %>
    姓名:<s:textfield name="username" value="%{#request.username}"></s:textfield>
<br/>
    <s:property value="%{'#request.username'}"/>

</body>
</html>
```

1.4 CRM 查询优化




150 <TD width=15><IMG src="${pageContext.request.contextPath }/images/new_024.jpg"
151 border=0></TD>
152 <TD align=middle width="100%"
153 background="${pageContext.request.contextPath }/images/new_025.jpg" height=15></TD>
154 <TD width=15><IMG src="${pageContext.request.contextPath }/images/new_026.jpg"
155 border=0></TD>
156 </TR>
157 </TBODY>
158 </TABLE>
159 </FORM>
160 </BODY>
161 </HTML>
162
html/BODY/FORM/TABLE/TBODY/TR/TD/TABLE/TBODY/TR/TD/TABLE/TBODY/siterator/#text
Writable
Smart Insert
94 : 37 : 3556