# 一、contextMap中的数据操作

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| root根：List   |  | | --- | | 元素1 | | 元素2 | | 元素3 | | 元素4 | | 元素5 | |
| contextMap：Map   |  |  | | --- | --- | | key | value | | application | Map   |  |  | | --- | --- | | key | value | | name | test | | | session | Map | | request | Map | | attr | Map | |

## 1、存数据：

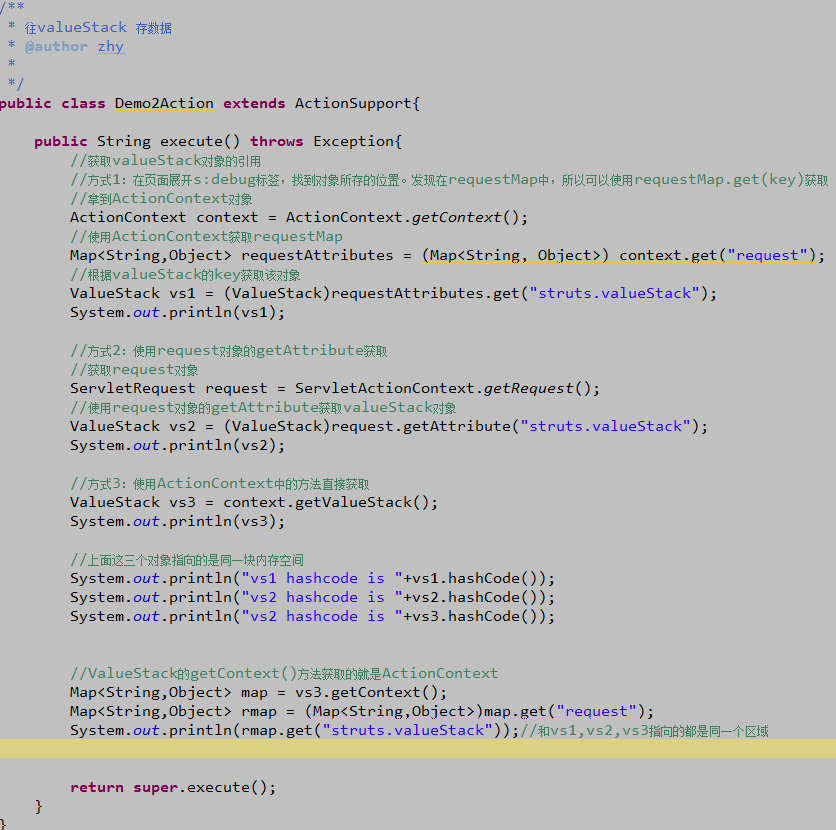
需要熟悉ActionContext和valueStack的API。框架为我们存数据。

### 1.1、利用ActionContext存数据

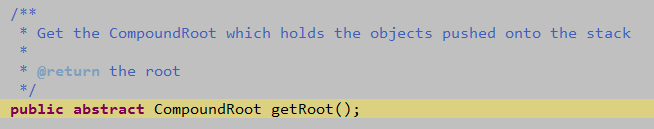


### 1.2、利用ValueStack存数据

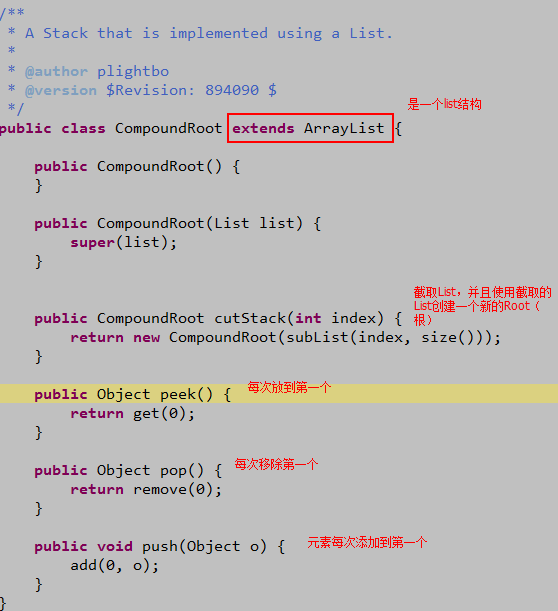
a、如何获取ValueStack：



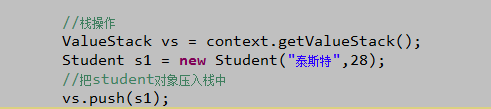
b、ValueStack中的getRoot()方法：



c、CompoundRoot是什么：



d、栈操作：

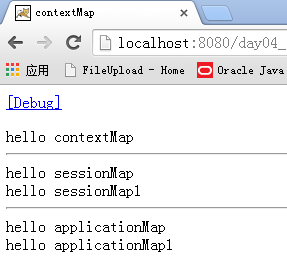


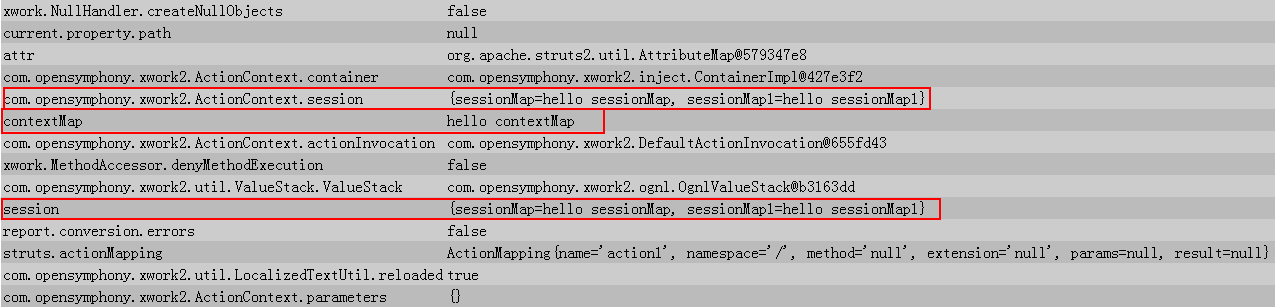
## 2、取数据：用Struts2的标签（OGNL表达式)在JSP上（用的最多）

使用OGNL表达式来去，struts2的OGNL表达式必须写在struts2标签中。

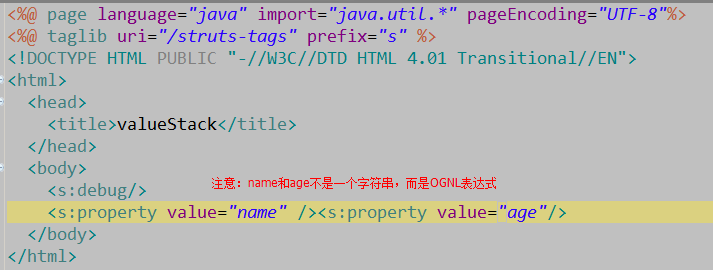
### 2.1、使用s:property取数据

**a、取contextMap中的数据，需使用#**

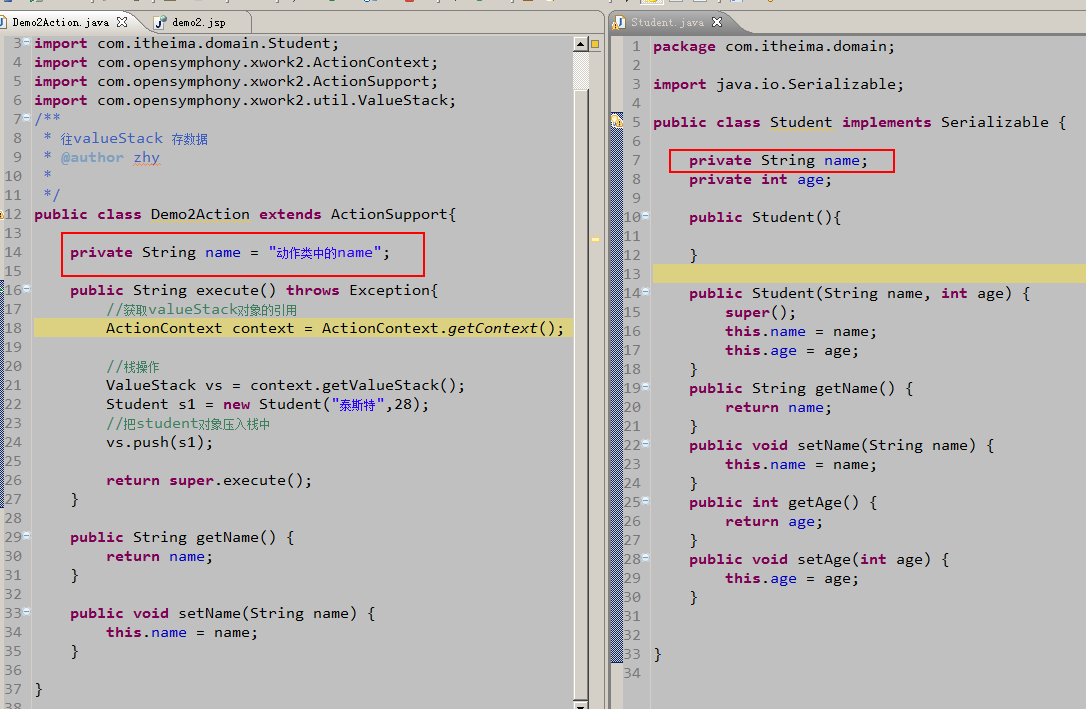


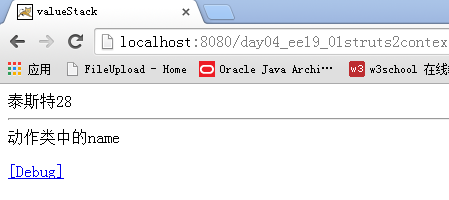
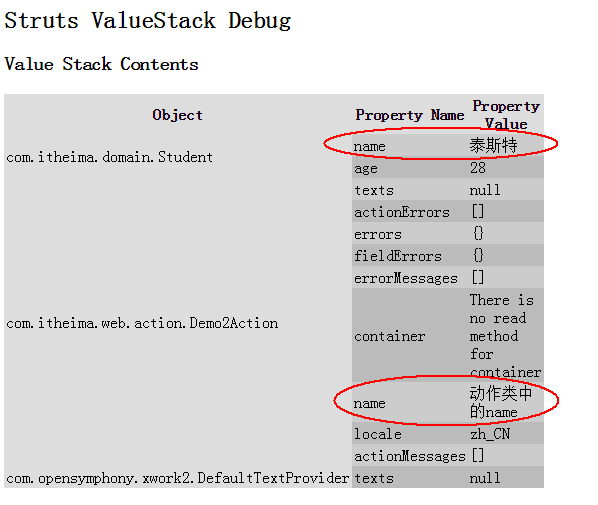
**b、取contextMap里面ValueStack中对象的属性：直接写属性名**

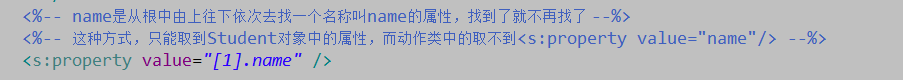
****

****

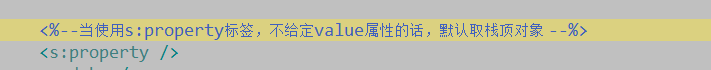
**如果遇有对象属性重名，可以通过OGNL表达式，选择查找的起始位置**

****

****

****

**特殊说明：当s:property不给定value属性时，默认取栈顶对象。**

****

****

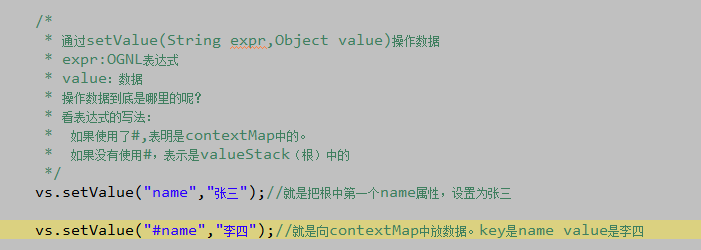
**OGNL的使用总结：**

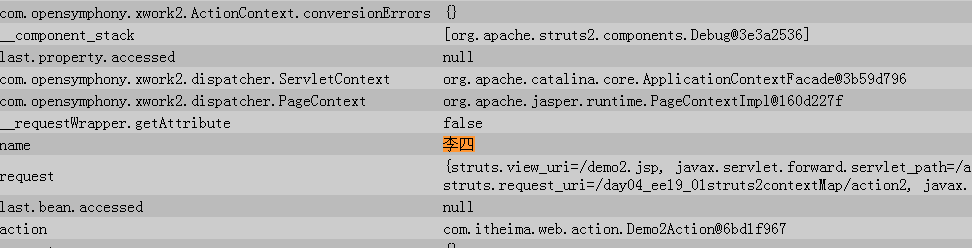
**1.取根中对象的属性，不使用#。**

**2.取contextMap中key的值，要使用#。**

### 2.2、ValueStack的其他方法：

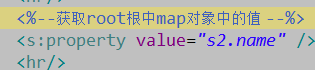
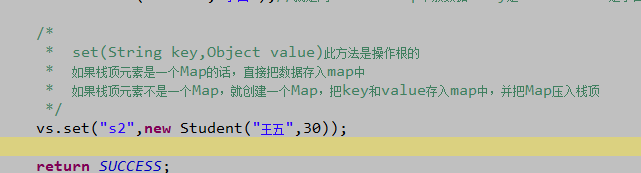
**a、setValue方法**





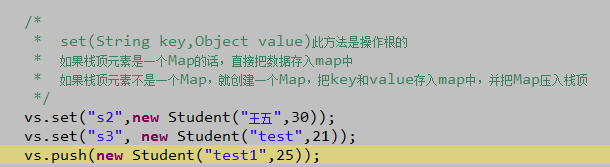
在Jsp页面上获取【李四】

**b、set方法**

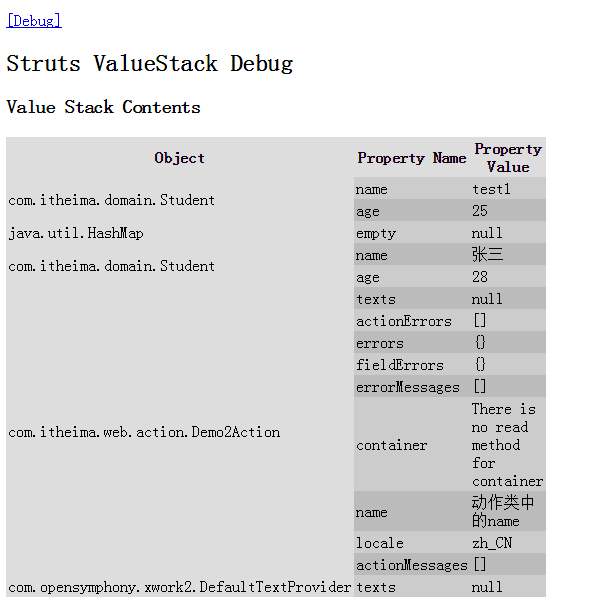
****

**细节问题：**

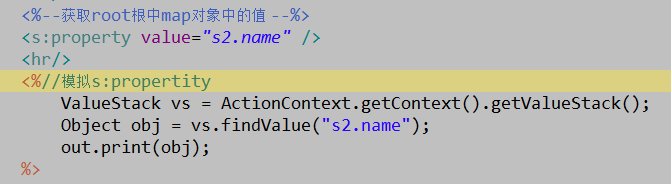
**当按照如下方式，往根中存放数据时，根中元素是什么顺序？**

****

**使用<s:debug/>标签查看：**

****

**c、findValue：我们在Jsp上调用的都是findValue**

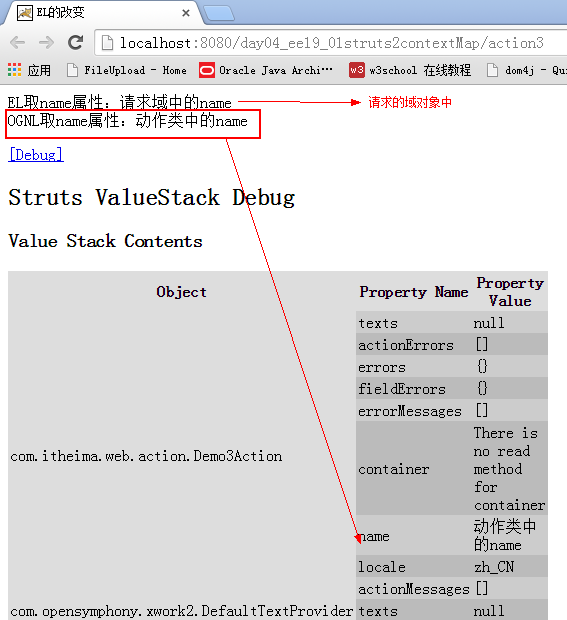
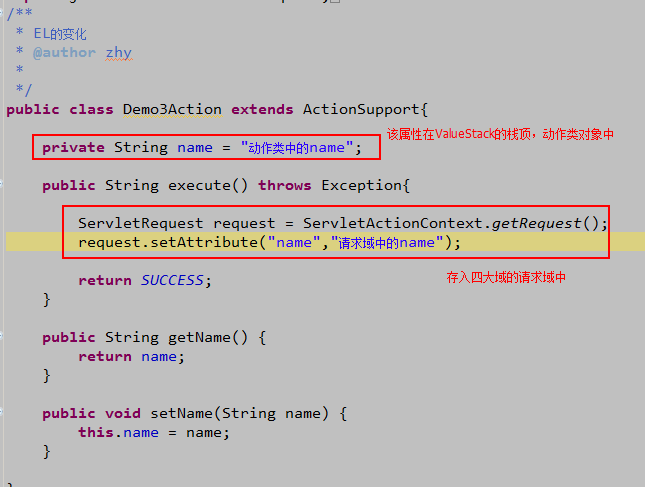
** **

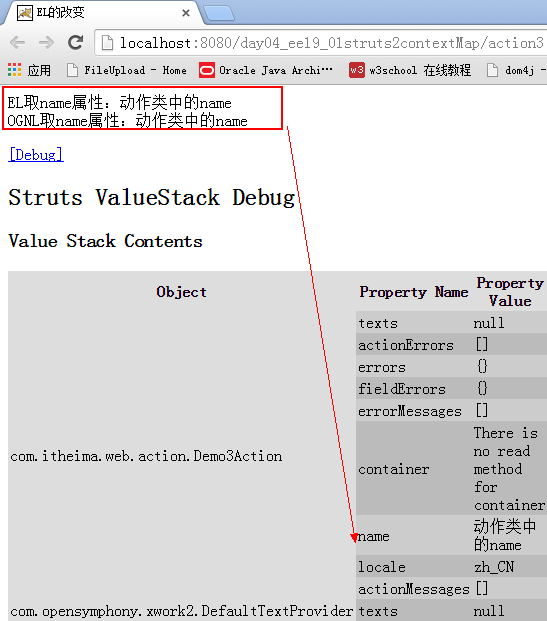
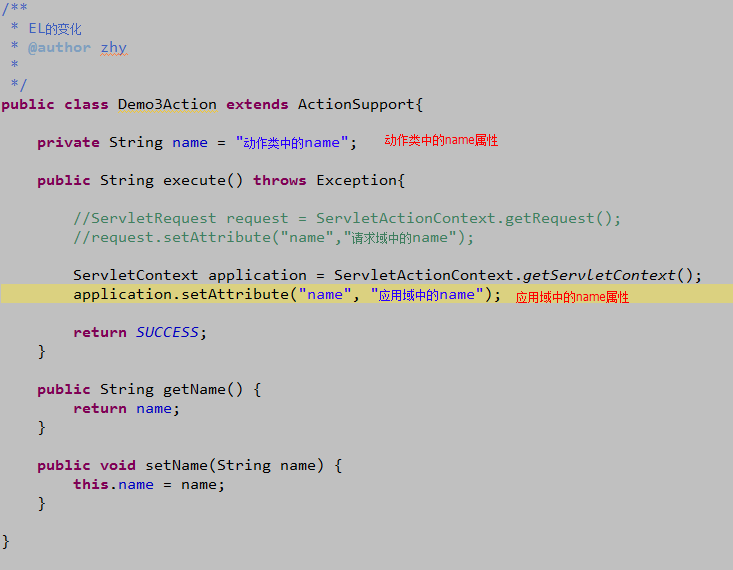
# 二、Struts2对EL的改变

## 1、Struts2中使用EL的问题：

**前提：**

我们应该知道，如果我们没有往值栈(根)中放入数据的话，那么我们的动作类默认是在值栈的栈顶。





问题：

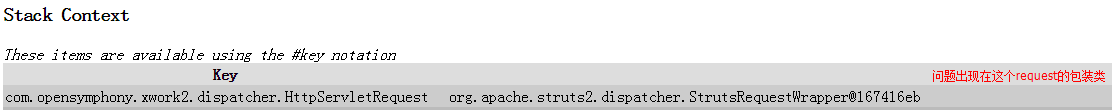
我们放到请求域中的属性，使用EL表达式取出来了。但是放到应用域中的属性，使用EL表达式没取出来。

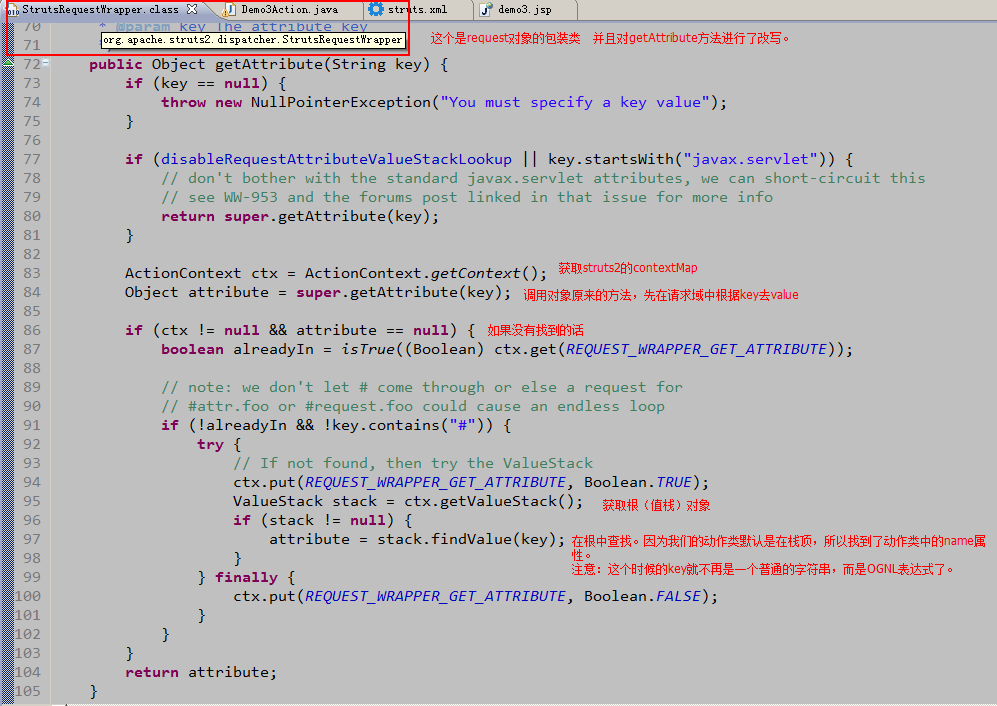
## 2、关于EL问题的分析：

**分析：**

我们知道EL表达式是从四大域对象中依次查找属性。搜索范围是由小到大。page Scope————>request Scope————>sessionScope————>application Scope

但是通过测试发现，搜索完request范围后就没有继续搜索，而是返回了ValueStack中栈顶对象name属性的值。



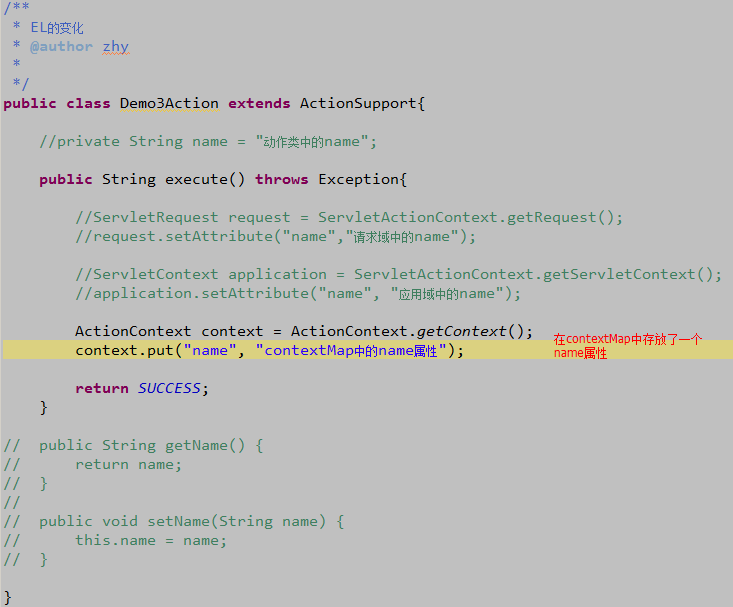
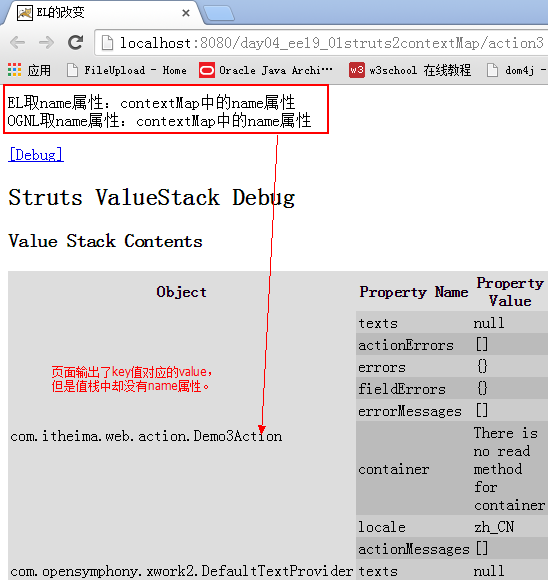


## 3、Struts2中EL查找顺序改变总结：

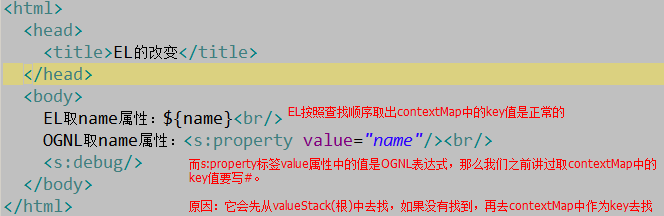
**EL表达式： page Scope————>request Scope————>sessionScope————>application Scope**

**OGNL表达式：page Scope————>request Scope————>valueStack（根中）————>contextMap————>sessionScope————>application Scope**

## 4、OGNL的特殊说明:

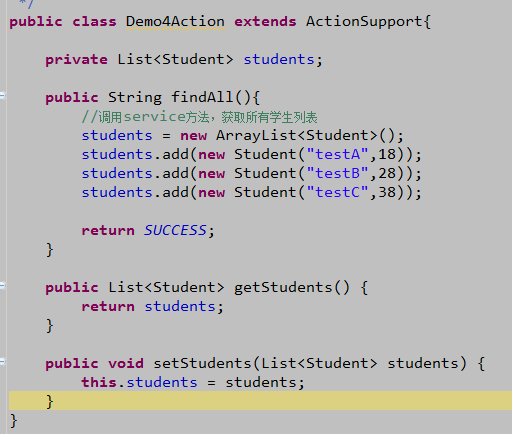
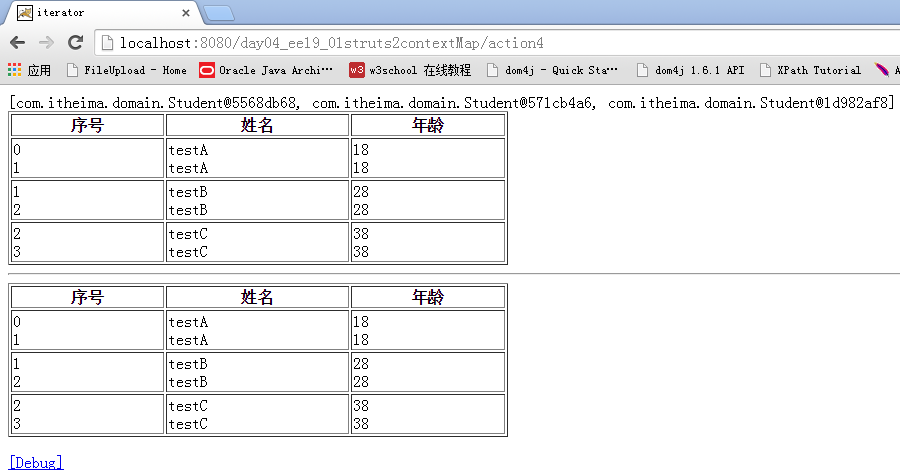
 

**注意：以下内容知道即可。**

****

# 三、OGNL配合通用标签的其他使用

## 1、iterator标签（很重要）

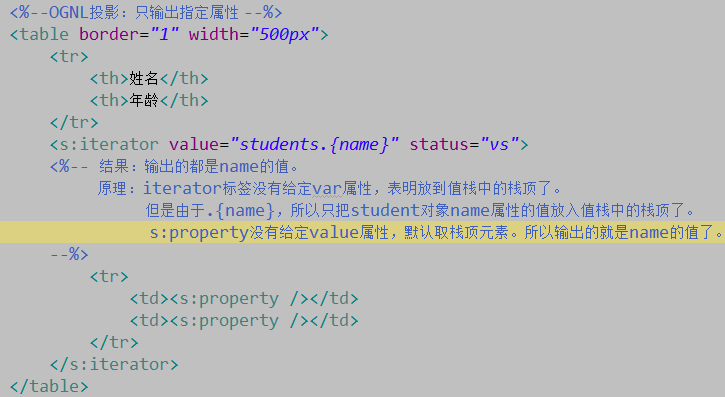


## 2、OGNL投影（了解）

### 2.1、使用过滤条件投影



### 2.2、投影指定属性



## 3、Struts2中#,$,%符号的使用(重要)

### 3.1、#

a、取contextMap中key时使用，例如<s:property value="#name" />

b、OGNL中创建Map对象时使用，例如：<s:radio list="#{'male':'男','female':'女'}" />

### 3.2、$

a、在JSP中使用EL表达式时使用，例如${name}

b、在xml配置文件中，编写OGNL表达式时使用，例如文件下载时，文件名编码。

struts.xml——>${@java.net.URLEncoder.encode(filename)}

### 3.3、%

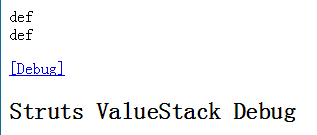
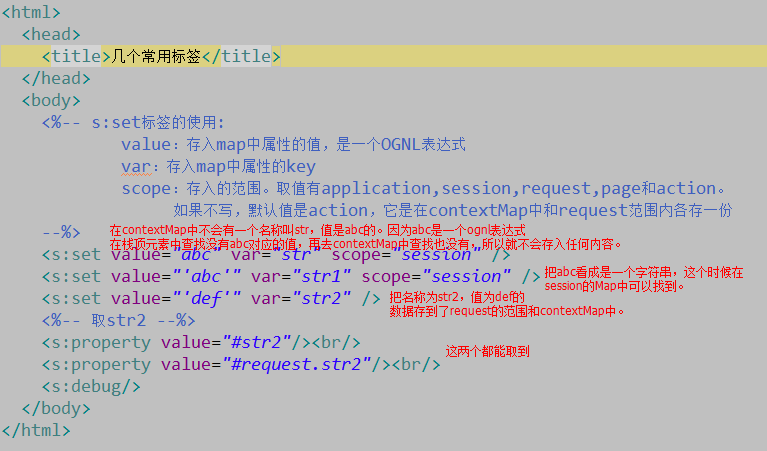
在struts2中，有些标签的value属性取值就是一个OGNL表达式，例如<s:property value="OGNL Expression" />

还有一部分标签，value属性的取值就是普通字 符串，例如<s:textfield value="username"/>，如果想把一个普通的字符串强制看成时OGNL，就需要使用%{}把字符串套起来。

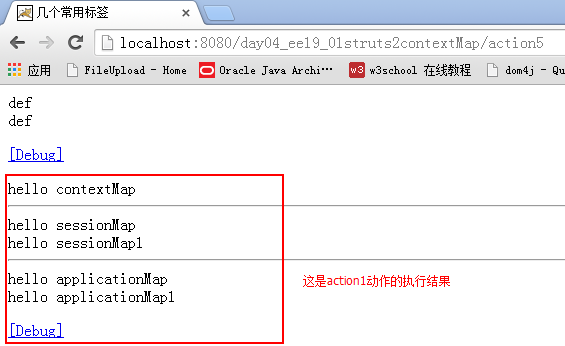
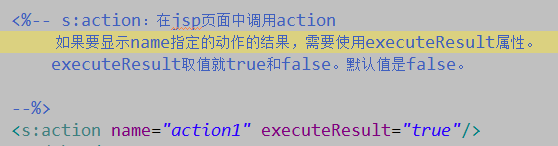
例如<s:textfield value="%{username}"/>。当然在<s:property value="%{OGNL Expression}" />也可以使用，但不会这么用。

## 4、其他标签

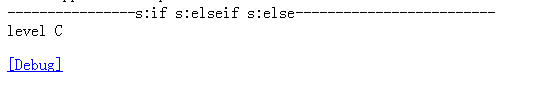
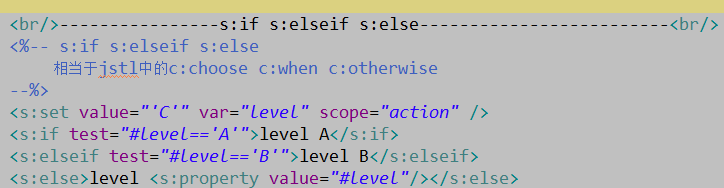
### 4.1、set标签



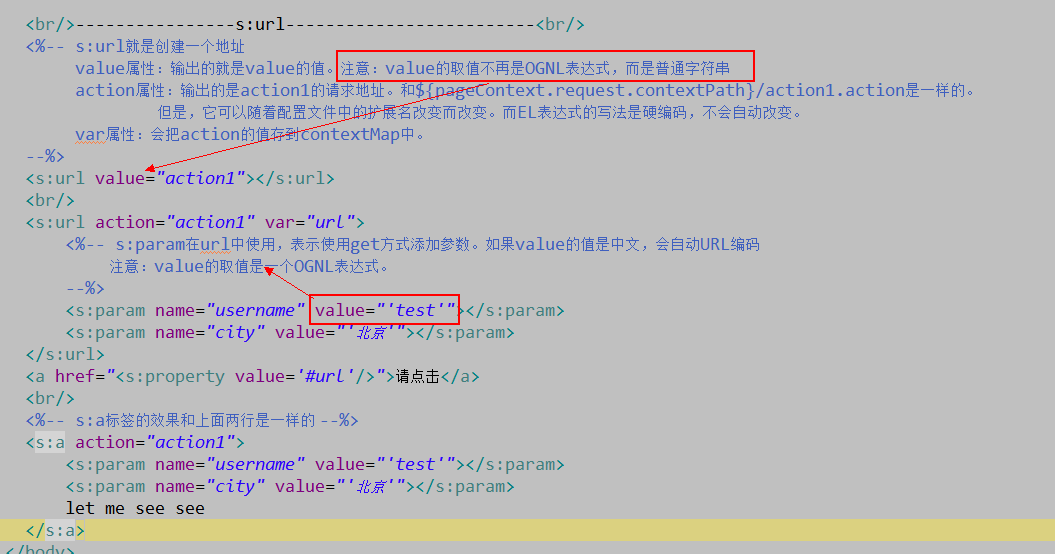
### 4.2、action标签



### 4.3、if标签，elseif标签 else标签



### 4.4、url和a标签（很有用）



# 四、Struts2的UI标签和主题

## 1、Struts2中UI标签的优势

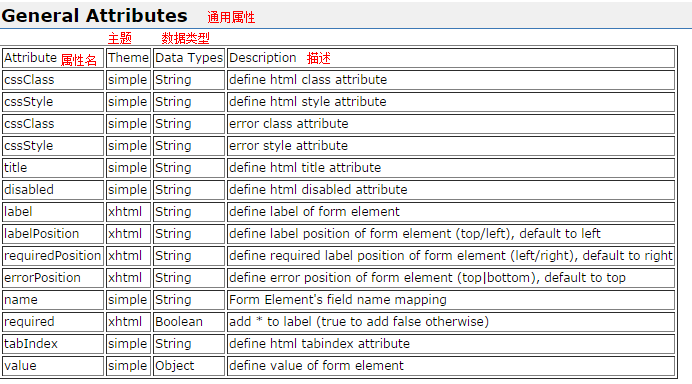
自动的数据回显和错误提示功能

自带的简单样式和排版

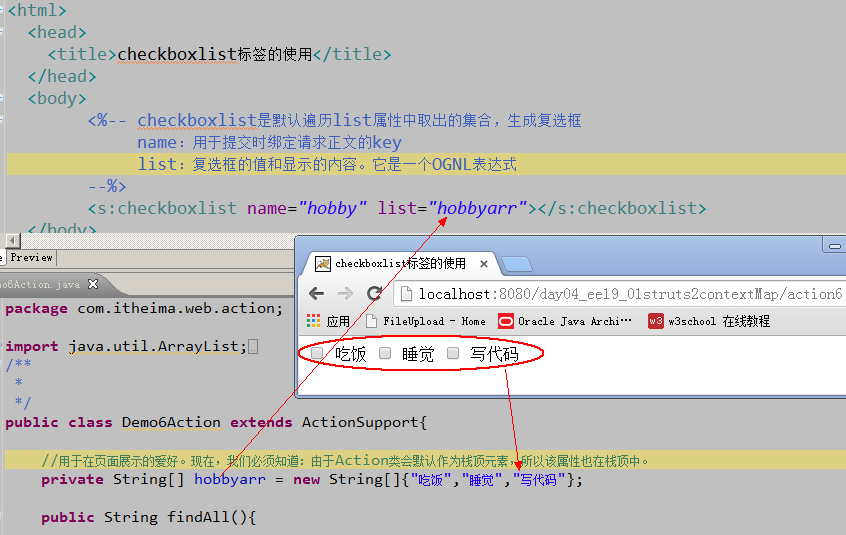
## 2、表单标签的通用属性

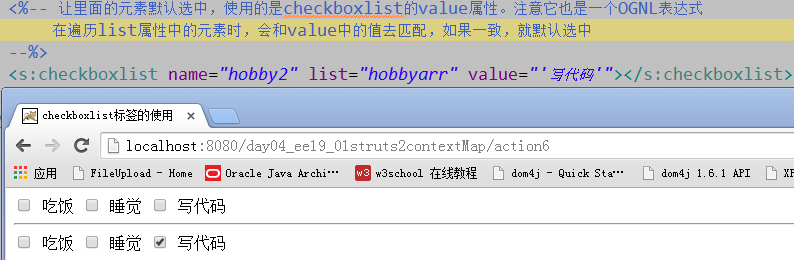
说明：UI标签中value的取值一般都是字符串。

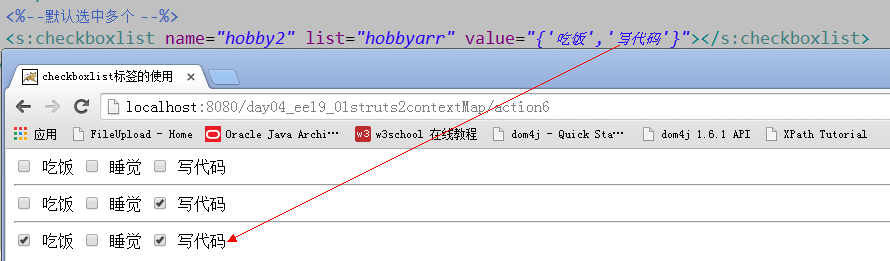
### 2.1、UI标签的通用属性

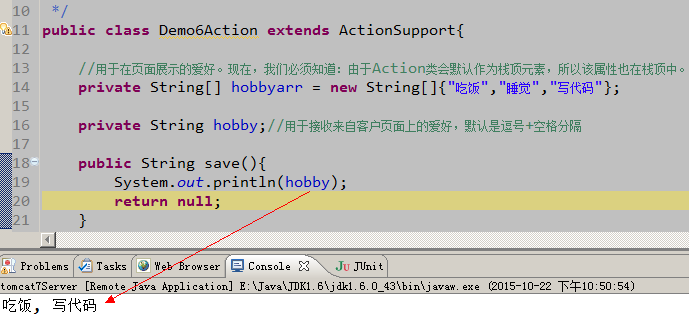


### 2.2、关于checkboxlist的使用：

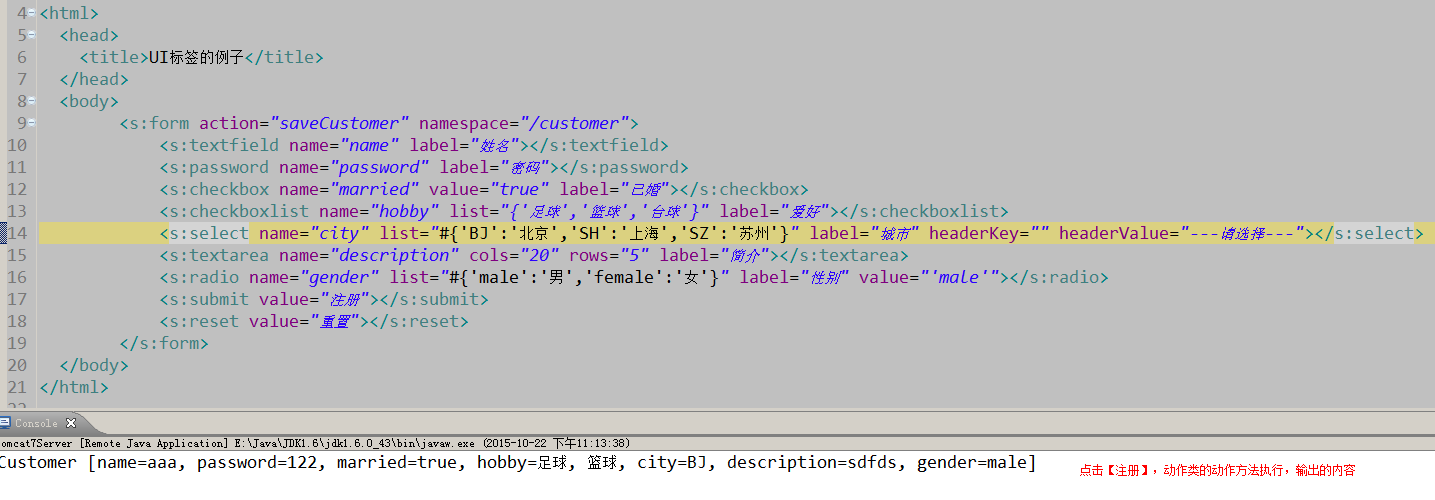
****

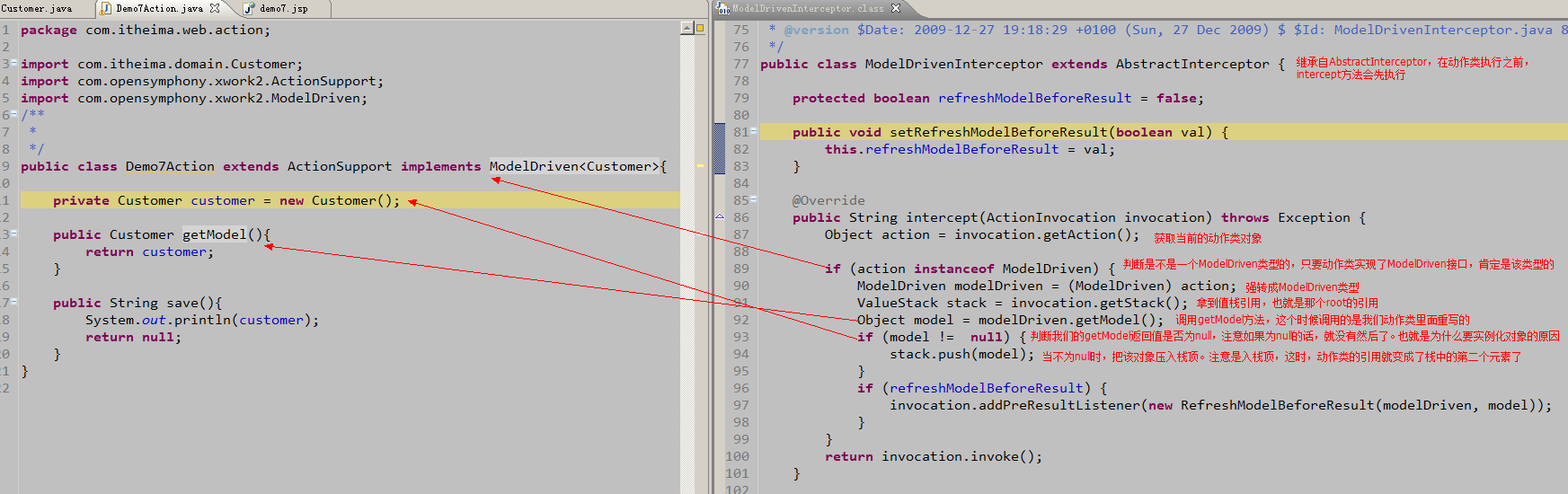
****

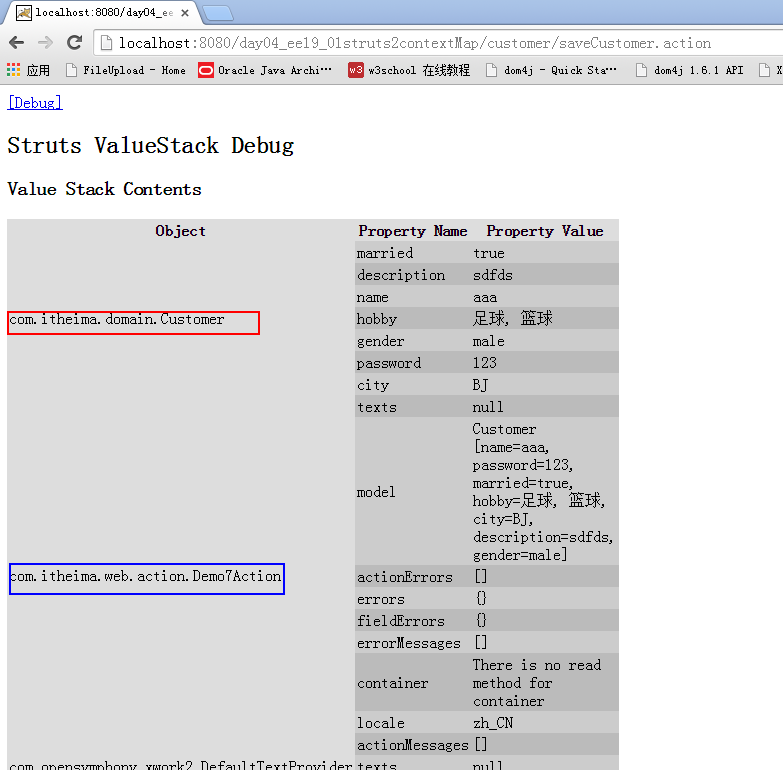
****

****

### 2.3、UI标签的小案例以及模型驱动的分析



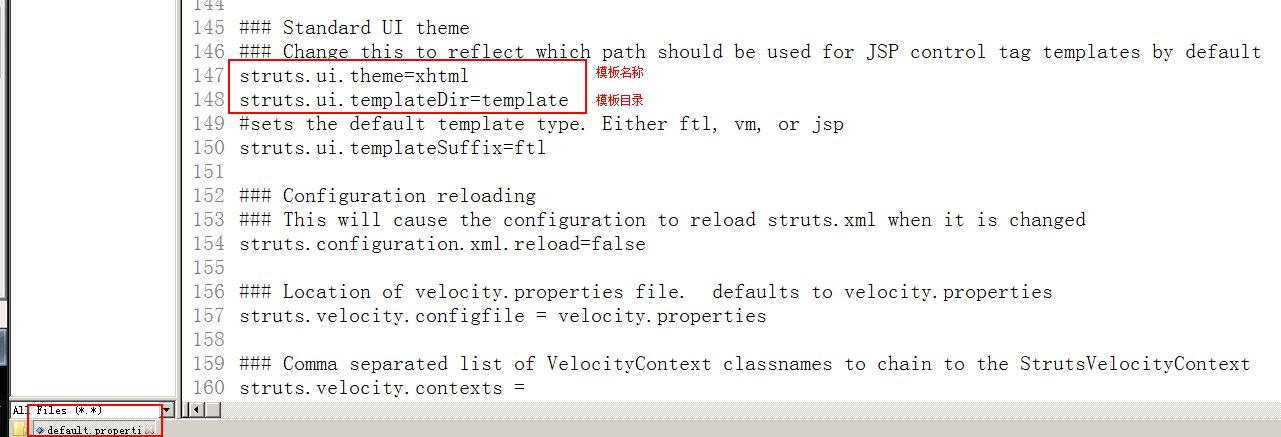


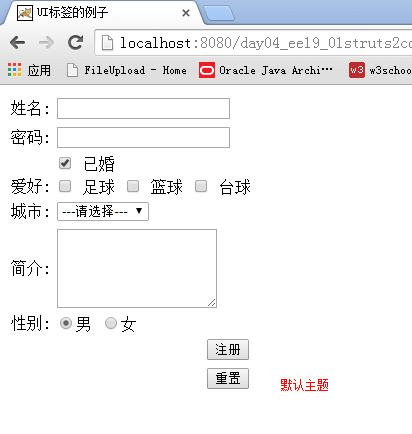
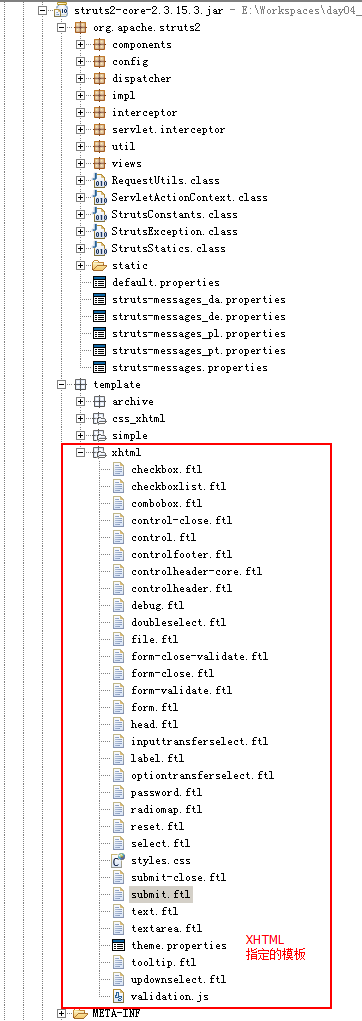


## 3、UI标签的模板（主题）

### 3.1、struts2中默认主题

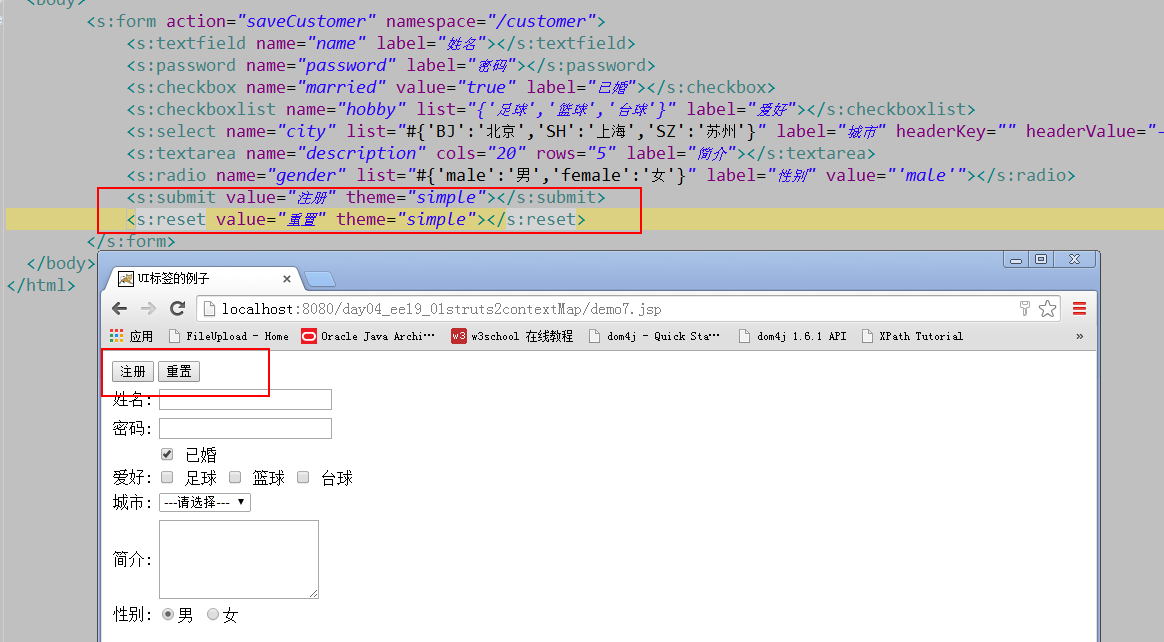
默认主题**的名称是XHTML，都是在struts的默认属性文件中定义着:default.properties**





### 3.2、更改默认主题

a、更改表单某个元素的默认主题：使用的是表单元素的theme属性。



b、更改表单所有主题：使用的是form标签的theme属性。

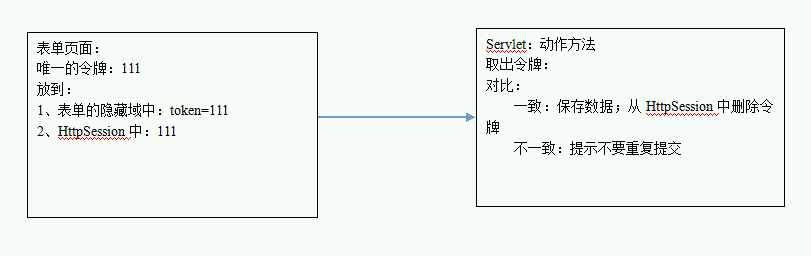


c、更改全站所有表单主题：是在struts.xml配置文件中，覆盖原有主题的设置。



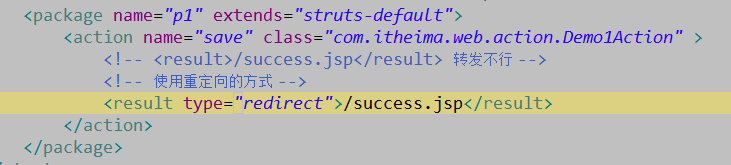
# 五、防止表单重复提交（拦截器）

## 1、回顾之前的解决办法：



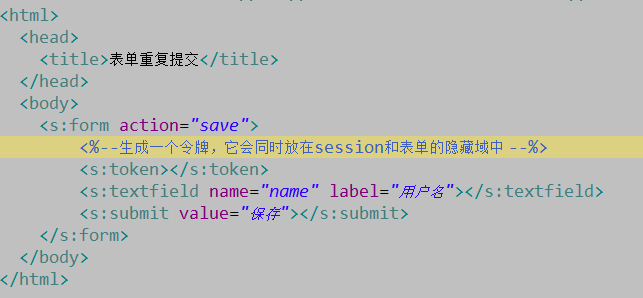
## 2、Struts2中的解决办法：

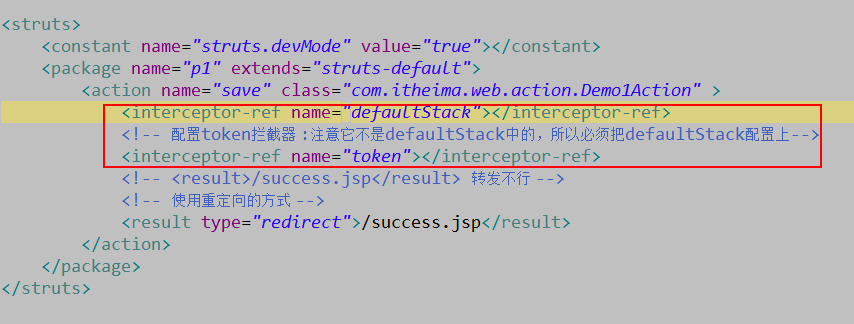
### 2.1、使用重定向



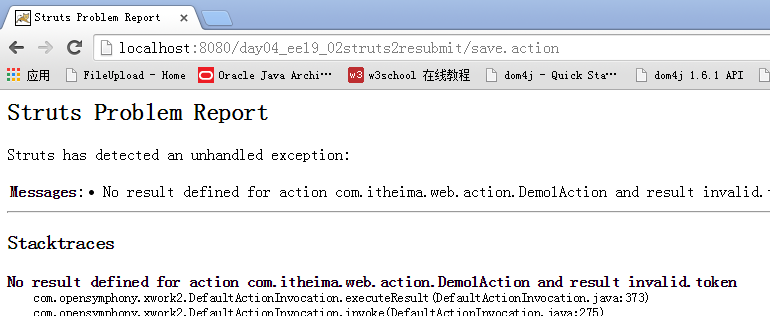
**遗留的问题：防不住后退，再提交。**

### 2.2、使用<s:token/>生成令牌配合token拦截器

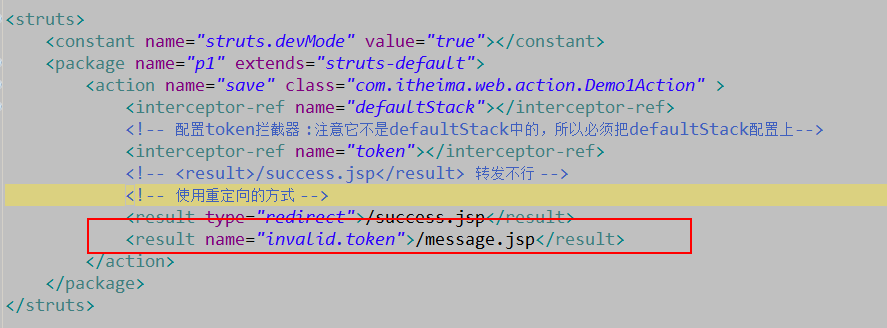




点击后退的时候，会提示：



**配置结果视图：**



**遗留的问题：此种解决方式，是产生了错误之后再告知用户，你错了。**

### 2.3、使用<s:token/>生成令牌配合tokensession拦截器

