直接使用Struts2框架来实现输入校验

1 手动实现校验

基本上都是在Action中完成校验的。**我们写的Action一定要继承自ActionSupport**

**public** **class** RegisterAction **extends** ActionSupport {

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

System.*out*.println("perform execute");

**return** *SUCCESS*;

}

@Override

**public** **void** validate() {

System.*out*.println("perform validate");

}

}

先做一个简单的实验，发现validate方法在execute方法之前执行，那么就可以在validate方法中完成校验逻辑。

**这里我们千万注意一个问题，在执行validate之前，一定要确保所有的用户请求参数值一定都要能完全无误的转换过来（无论是使用Struts2内置转换器还是自定义转换器），否则的话，在执行validate方法（如果有重写了validate方法，不管有没有在validate方法中添加错误）后系统就会跳转到INPUT逻辑视图名所指向的视图资源！！！！！！所以说只用在确保类型转换成功的情况下执行输入校验才有意义！！！！！！在执行validate方法时，只要我们在该方法中添加了Action或Field级别的错误，那么Action执行就此打住，直接跳转到INPUT指向的视图资源。否则的话，所有验证都通过了才会执行execute方法。**

**public** **class** RegisterAction **extends** ActionSupport {

**private** String name;

**private** **int** age;

**private** Date birth;

**…省略了Getters和Setters方法**

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

System.*out*.println("perform execute");

**return** *SUCCESS*;

}

@Override

**public** **void** validate() {

**this**.addFieldError("name", "name is error");

**this**.addFieldError("age", "age is error");

**this**.addFieldError("birth", "birth is error");

}

}

这时候我们还要在Action中配置INPUT结果

<action name=*"register"* class=*"RegisterAction"*>

<result>/index.jsp</result>

<result name=*"input"*>/index.jsp</result>

</action>

在index.jsp页面中加入：

<s:fielderror></s:fielderror>

除了将验证逻辑封装在validate方法中，还可以放在validateXxx方法中

为了能调用validateXxx方法，首先要调用Action中的xxx方法（Xxx和xxx只有首字母是否大小写的区别，其他一模一样），为了让Action一获取用户请求就执行xxx方法，首先要在action中配置method为xxx 如：

**public** **class** RegisterAction **extends** ActionSupport {

**private** String name;

**private** **int** age;

**private** Date birth;

…省略了Getters和Setters方法

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

System.*out*.println("perform execute");

**return** *SUCCESS*;

}

@Override

**public** **void** validate() {

System.*out*.println("validate invoked");

**this**.addFieldError("name", "name error");

}

**public** **void** validate**Reg**(){

System.*out*.println("validateReg invoked");

**this**.addFieldError("name", "name is error!!!!");

}

**public** String **reg**(){

System.*out*.println("reg invoked");

**return** *SUCCESS*;

}

**这里的调用顺序是：validateReg()方法 validate()方法 reg()方法（只有前两个方法验证通过了才执行）**

**使用配置文件来实现输入校验**

我们只要在Action中定义与用户请求参数对应的属性，并实现Getters和Setters方法

在Action所在的包下面定义一个校验规则文件，文件名是：Action名字-validation.xml

校验文件的内容是：

<field name=*"****用户请求参数名****"*>

<field-validator type=*"****规则类型****"*>

<param name=*"****参数名****"*>***参数值***</param>

<message>***错误消息***</message>

</field-validator>

</field>

常用的验证规则：

字符串类型的验证：

type param name value

非空：requiredstring trim true || false

正则：regex expression <![CDATA[( )]]>

数字类型的验证：

数值范围：int min number

max number

日期类型的验证：

日期范围：date min yyyy-MM-dd

max yyyy-MM-dd

**注意非常重要的一点，在Struts2中数据校验是建立在类型转换的基础上的！！！对于任何表单域是空的情况，提交到服务器端的参数值都是空字符串！！！这样就存在类型转换失败的问题，因此说Struts2校验机制无法校验用户输入为空的情况！！！！这需要我们手动验证！！！！**

使用这种验证方式发现一个问题，就是日期类型的表单域无法验证输入为空的情况，还需要使用其他的验证方法。

使用国际化资源文件来显示错误信息

可以在上面的<message>标签中使用这样的形式：

<message key=”name” />的形式

这里的key所表示的就是国际化资源文件中的name，这样我们只要在国际化资源文件中配置错误信息了

使用客户端校验

只需要将提交表单用Struts2标签来改写就行了，修改<s:form action=”register” validate=”true”>

验证规则文件不要做任何变化

Action中验证逻辑的分离

一个Action中可以有多个类似于execute的方法，每个方法都对应一个校验规则文件，想要调用某种校验逻辑的时候可以在struts.xml文件中为action指定method属性，这时候的校验规则文件的命名就要注意了，Action名字-方法名-validation.xml，当调用method指定的方法时，就会综合Action名字-validation.xml和Action名字-方法名-validation.xml这两个校验规则文件来验证，一般是先验证前面的文件再验证后者

如：MyAction-validation.xml文件的主要校验规则：

<field name=*"name"*>

<field-validator type=*"requiredstring"*>

<param name=*"trim"*>true</param>

<message>姓名不能为空</message>

</field-validator>

<field-validator type=*"regex"*>

<param name=*"expression"*><![CDATA[(\w{4,8})]]></param>

<message>长度在4到8之间</message>

</field-validator>

</field>

<field name=*"age"*>

<field-validator type=*"int"*>

<param name=*"min"*>1</param>

<param name=*"max"*>150</param>

<message>年龄必须在1到150之间</message>

</field-validator>

</field>

而MyAction-login-validation.xml文件的验证规则如下：

<field name=*"name"*>

<field-validator type=*"fieldexpression"*>

<param name=*"expression"*><![CDATA[(name == "hahaha" )]]></param>

<message>姓名不正确</message>

</field-validator>

</field>

校验规则如下：

Action名字-方法名-validation.xml中有和Action名字-validation.xml相同的校验规则，前者会覆盖后者。还有就是MyAction 继承自 BaseAction，如果MyAction中调的是execute方法，则搜索校验规则的顺序是：BaseAction-validation.xml MyAction-validation.xml

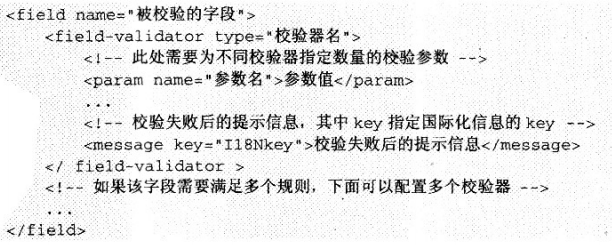
如果MyAction中调的是xxx方法，则搜索校验规则的顺序是：BaseAction-validation.xml

BaseAction-xxx-validation.xml MyAction-validation.xml MyAction-xxx-validation.xml

总之就是按照这个顺序查找，找不到也无所谓。

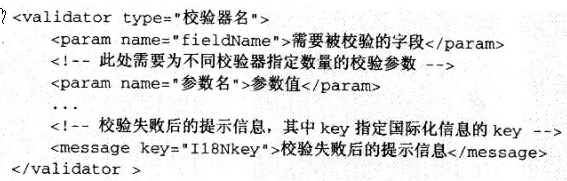
校验器风格

字段校验器风格



注：截图引用自《轻量级Java EE企业应用实践（第二版）》

非字段校验器风格



注：截图引用自《轻量级Java EE企业应用实践（第二版）》

短路校验

表示的是对同一个表单域有好几个校验规则，如果第一个校验失败，后面的就不会再验证了

这要在配置：

<field-validator type=" " short-circuit=”true”></…>

现在我有一个疑问就是，我在表单中的form中加了validate=”true”，这就意味着使用的是客户端校验，但是有些校验器在客户端是不支持的，怎么办呢？我的想法是首先第一次提交表单时，同时使用客户端校验和服务器端校验，接下来进行客户端校验，等客户端校验都完成了，才提交到服务器端校验吧。

Struts2中的内置校验器：

1. 必填校验器 required

发现一个很棘手的问题，对于**数值类型**的文本域什么都不填，使用必填校验器第一次提交时根本就不会验证！！！！

1. 邮件校验器 email
2. Visitor校验器？？？
3. 转换校验器 conversion repopulatefield

true || false

…用的时候查阅资料，我也懒得列全了

经过上面的学习和研究，发现下面一些问题和解决办法：

**所有的表单域，所有的文本域我们都使用js来验证非空的情况，对于数值型和日期型的文本域都要确保类型、格式的正确，这也使用js来验证，对于其他的所有情况都交给Struts2框架来验证！！！！！！！！！！！！！！！！！！！！！！！！！！！！还有一点就是要把js验证和Struts2验证的错误显示风格统一起来，让我们的js验证的错误显示风格服从于Struts2的xhtml主题风格，那最好就是使用jQuery来实现了，今天又发现一个控制用户输入类型和格式的绝佳方法，拦截器！！！！**