# Validate（校验）是用于检验传入数据的

Struts2 中Validate（校验）的实现有三种方法

# 第一种：

通过在方法体中直接编写校验逻辑（不建议使用，破坏编码风格）

# 第二种：

通过在方法类中重写ActonSupport中的validate()(前提是方法类得继承ActionsSupport)

## Note：

validate方法实质是通过调用拦截器来实现对数据的校验的

Validate 有两种校验拦截模式

如不通过校验，拦截器中会返回"input",因此struts.xml中如无相应的result设置，会出现错误

## validate的总拦截：

在方法类中直接重写validate()方法，该方法会在action调用任何方法前激发拦截器，对传入数据进行校验

例：

/\*\*

\* 在action调用任何方法前执行

\*

\* 如不通过校验，拦截器中会返回"input",

\* 因此struts.xml中如无相应的result设置，会出现错误

\*/

@Override

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

**if**(user.getUsername()==**null** || user.getUsername().length()<6){

addFieldError("user.username", "用户名不能为空或低于六个字节");

}

}

## validate的指定方法拦截：

定义的指定方法拦截只有action调用所指定的方法，才会提前激发拦截器，对传入数据进行校验

例：

/\*\*

\* 在 action调用select()前执行

\*

\* 如不通过校验，拦截器中会返回"input",

\* 因此struts.xml中如无相应的result设置，会出现错误

\*

\*/

**public** **void** validateDelete(){

**if**(user.getUsername()==**null** || user.getUsername().length()<6){

addFieldError("User.Username", "输入的用户名不符合命名规则");

}

}

Note：

使用validate的指定方法拦截时，必须遵循struts2的命名规则，规则为 validate+方法名(),如要拦截方法类中delete()方法，则指定方法的validate拦截必须命名为validateDelete()

# 第三种：

通过xml文件配置

## Note：

配置校验的xml文件必须与其所校验的方法类在同一个package下

如不通过校验，拦截器中会返回"input",因此struts.xml中如无相应的result设置，会出现错误

有两种校验拦截模式

## validate的总拦截：

命名规则： ClassName-validation.xml

例如想要设置在UserAction类上的总拦截，则其xml文件名必须为UserAction-validation.xml

## validate的指定方法拦截：

命名规则： ClassName-validation-methodName.xml

例如想要设置在UserAction类的delete()方法上的拦截，则其xml文件名必须为UserAction-validation-delete.xml

## Struts2常用的验证器：

required:必填校验器  
requiredstring:必填字符串校验器  
int:整数校验器  
double:双精度浮点数校验器  
date:日期校验器  
expression:表达式校验器  
fieldexpression:字段表达式校验器  
email:电子邮件校验器  
url:网址校验器  
visitor:Visitor校验器  
conversion:转换校验器  
stringlength:字符串长度校验器  
regex:正则表达式校验器

用法可以看框架的源码，所属包为com.opensymphony.xwork2.validator.validators

## 校验xml的写法示例：

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

<!DOCTYPE validators PUBLIC

"-//Apache Struts//XWork Validator 1.0.3//EN"

"http://struts.apache.org/dtds/xwork-validator-1.0.3.dtd">

<validators>

<field name=*"user.name"*>

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

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

<message>xml校验：名字不可为空</message>

</field-validator>

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

<param name=*"minLength"*>6</param>

<param name=*"maxLength"*>11</param>

<message>姓名长度应在5～11个字符间</message>

</field-validator>

</field>

<field name=*"user.age"*>

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

<message>xml校验：年龄不可为空</message>

</field-validator>

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

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

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

<message>xml校验：请输入符合常识的年龄</message>

</field-validator>

</field>

<field name=*"user.email"*>

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

<message>xml校验：邮箱不可为空</message>

</field-validator>

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

<message>xml校验：请正确输入邮箱</message>

</field-validator>

</field>

<field name=*"user.phone"*>

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

<message>xml校验：手机号码不可为空</message>

</field-validator>

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

<param name=*"expression"*>

<![CDATA[((13[4,5,6,7,8,9]|15[0,8,9,1,7]|188|187)\\d{8})]]>

</param>

<param name=*"caseSensitive"*>false</param>

<message>xml校验：请正确输入手机号码</message>

</field-validator>

</field>

<field name=*"user.password"*>

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

<message>请填写密码</message>

</field-validator>

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

<param name=*"expression"*>

<![CDATA[(user.getPassword().length() < 6 || user.getPassword().length() > 15]]>

</param>

<message>xml校验：请输入6～15位长度的密码</message>

</field-validator>

</field>

</validators>

# 在jsp中获取传入的校验错误信息：

Prefix设置：

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

获取校验错误信息：

<s:fielderror/>

<s:actionerror/>