输入数据校验

输入校验

- 在struts2中，我们可以实现对action的所有方法进行校验或者对action的指定方法进行校验。

- 对于输入校验struts2提供了两种实现方法:
- 1. 采用手工编写代码实现。
- 2. 基于XML配置方式实现。

手工编写代码实现对action中所有方法输入校验

- 通过重写validate() 方法实现， validate()方法会校验action中所有与execute方法签名相同的方法。
- 当某个数据校验失败时，我们应该调用addFieldError()方法往系统的fieldErrors添加校验失败信息（为了使用addFieldError()方法，action可以继承ActionSupport ），如果系统的fieldErrors包含失败信息，struts2会将请求转发到名为input的result。
- 在input视图中可以通过<s:fielderror/>显示失败信息。

- 验证失败后，请求转发至input视图：
- <result name="input">/register.jsp</result>

- 在addUser.jsp页面中使用<s:fielderror/>显示失败信息。

对action中所有方法输入校验示例

```
public void validate() {
	if (this.username == null || "".equals(username)) {
		this.addFieldError("username", "用户名不能为空");
	}

	if (this.mobile == null || !this.mobile.matches("^\\d{11}$")) {
		this.addFieldError("mobile", "手机号码格式不正确");
	}

	if (this.age == null || "".endsWith(this.age)) {
		this.addFieldError("age", "年龄不能为空");
	} else if (!this.age.matches("^\\d{1,3}$")) {
		this.addFieldError("age", "年龄是1~3位正整数");
	}
}
```

手工编写代码实现对action指定方法输入校验

- 通过validateXxx()方法实现， validateXxx()只会校验action中方法名为Xxx的方法。其中Xxx的第一个字母要大写。
- 当某个数据校验失败时，我们应该调用addFieldError()方法往系统的fieldErrors添加校验失败信息（为了使用addFieldError()方法，action可以继承ActionSupport ），如果系统的fieldErrors包含失败信息，struts2会将请求转发到名为input的result。
- 在input视图中可以通过<s:fielderror/>显示失败信息。

对action指定方法输入校验示例

```
//单独校验update方法
public void validateUpdate() {
        if (this.username == null || "".equals(username)) {
                this.addFieldError("username", "用户名不能为空");
        }
        if (this.mobile == null || !this.mobile.matches("^\\d{11}$")) {
                this.addFieldError("mobile", "手机号码格式不正确");
        }
        if (this.age == null || "".endsWith(this.age)) {
                this.addFieldError("age", "年龄不能为空");
        } else if (!this.age.matches("^\\d{1,3}$")) {
                this.addFieldError("age", "年龄是1~3位正整数");
        }
}
```

输入校验的流程

1。类型转换器对请求参数执行类型转换，并把转换后的值赋给action中的属性。

2。如果在执行类型转换的过程中出现异常，系统会将异常信息保存到ActionContext，conversionError拦截器将异常信息添加到fieldErrors里。不管类型转换是否出现异常，都会进入第3步。

3。系统通过反射技术先调用action中的validateXxx()方法，Xxx为方法名。

4。再调用action中的validate()方法。

5。经过上面4步，如果系统中的fieldErrors存在错误信息（即存放错误信息的集合的size大于0)，系统自动将请求转发至名称为input的视图。如果系统中的fieldErrors没有任何错误信息，系统将执行action中的处理方法。

基于XML配置方式实现对action的所有方法进行输入校验

使用基于XML配置方式实现输入校验时，Action也需要继承ActionSupport，并且提供校验文件，校验文件和action类放在同一个包下，文件的取名格式为：ActionClassName-validation.xml，其中ActionClassName为action的简单类名，-validation为固定写法。如果Action类为UserAction，那么该文件的取名应为：UserAction-validation.xml。下面是校验文件的模版：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE validators PUBLIC "-//OpenSymphony Group//XWork Validator 1.0.3//EN"
    "http://www.opensymphony.com/xwork/xwork-validator-1.0.3.dtd">
<validators>
   <field name="username">
      <field-validator type="requiredstring">
         <param name="trim">true</param>
         <message>用户名不能为空!</message>
      </field-validator>
   </field>
</validators>
```

<field>指定action中要校验的属性，<field-validator>指定校验器，上面指定的校验器requiredstring是由系统提供的，系统提供了能满足大部分验证需求的校验器，这些校验器的定义可以在xwork-2.x.jar中的com.opensymphony.xwork2.validator.validators下的default.xml中找到。

<message>为校验失败后的提示信息,如果需要国际化，可以为message指定key属性，key的值为资源文件中的key。

在这个校验文件中，对action中字符串类型的username属性进行验证，首先要求调用trim()方法去掉空格，然后判断用户名是否为空。

基于XML配置方式对指定action方法实现输入校验

当校验文件的取名为ActionClassName-validation.xml时，会对 action中的所有处理方法实施输入验证。如果你只需要对action中的某个action方法实施校验，那么，校验文件的取名应为:ActionClassName-ActionName-validation.xml，其中ActionName为struts.xml中action的名称。例如：在实际应用中，常有以下配置：

```
<action name="user_*" class="cn.itcast.action.UserAction" method="{1}“ >
    <result name="success">/WEB-INF/page/message.jsp</result>
    <result name="input">/WEB-INF/page/addUser.jsp</result>
</action>
```

UserAction中有以下两个处理方法：

```
public String add() throws Exception{
   ....
}
public String update() throws Exception{
   ....
}
```

要对add()方法实施验证，校验文件的取名为： UserAction-user_add-validation.xml

要对update()方法实施验证，校验文件的取名为： UserAction-user_update-validation.xml

struts2提供的校验器列表

- 系统提供的校验器如下：
- required (必填校验器,要求field的值不能为null)
- requiredstring (必填字符串校验器,要求field的值不能为null,并且长度大于0,默认情况下会对字符串去前后空格)
- stringlength(字符串长度校验器,要求field的值必须在指定的范围内,否则校验失败,minLength参数指定最小长度,maxLength参数指定最大长度,trim参数指定校验field之前是否去除字符串前后的空格)
- regex(正则表达式校验器,检查被校验的field是否匹配一个正则表达式.expression参数指定正则表达式,caseSensitive参数指定进行正则表达式匹配时,是否区分大小写,默认值为true)
- int(整数校验器,要求field的整数值必须在指定范围内,min指定最小值,max指定最大值)
- double(双精度浮点数校验器,要求field的双精度浮点数必须在指定范围内,min指定最小值,max指定最大值)
- fieldexpression(字段OGNL表达式校验器,要求field满足一个ognl表达式,expression参数指定ognl表达式,该逻辑表达式基于ValueStack进行求值,返回true时校验通过,否则不通过)
- email(邮件地址校验器,要求如果field的值非空,则必须是合法的邮件地址)
- url(网址校验器,要求如果field的值非空,则必须是合法的url地址)
- date(日期校验器,要求field的日期值必须在指定范围内,min指定最小值,max指定最大值)
- conversion(转换校验器,指定在类型转换失败时,提示的错误信息)
- visitor(用于校验action中的复合属性,它指定一个校验文件用于校验复合属性中的属性)
- expression(OGNL表达式校验器,expression参数指定ognl表达式,该逻辑表达式基于ValueStack进行求值,返回true时校验通过,否则不通过,该校验器不可用在字段校验器风格的配置中)

校验器的使用例子

required 必填校验器

```
<field-validator type="required">
      <message>性别不能为空!</message>
</field-validator>
```

requiredstring 必填字符串校验器

```
<field-validator type="requiredstring">
      <param name="trim">true</param>
      <message>用户名不能为空!</message>
</field-validator>
```

stringlength：字符串长度校验器

```
<field-validator type="stringlength">
     <param name="maxLength">10</param>
     <param name="minLength">2</param>
     <param name="trim">true</param>
     <message><![CDATA[产品名称应在2-10个字符之间]]></message>
</field-validator>
```

校验器的使用例子

email：邮件地址校验器

```
<field-validator type="email">
    <message>电子邮件地址无效</message>
</field-validator>
```

regex：正则表达式校验器

```
<field-validator type="regex">
    <param name="expression"><![CDATA[^1[358]\d{9}$]]></param>
    <message>手机号格式不正确!</message>
</field-validator>
```

校验器的使用例子

int：整数校验器

```
<field-validator type="int">
    <param name="min">1</param>
    <param name="max">150</param>
    <message>年龄必须在1-150之间</message>
</field-validator>
```

字段OGNL表达式校验器

```
<field name="imagefile">
    <field-validator type="fieldexpression">
        <param name="expression"><![CDATA[imagefile.length() <= 0]]></param>
        <message>文件不能为空</message>
    </field-validator>
</field>
```

基于XML校验的一些特点

- 当为某个action提供了ActionClassName-validation.xml和ActionClassName-ActionName-validation.xml两种规则的校验文件时，系统按下面顺序寻找校验文件：
- 1。AconClassName-validation.xml
- 2。ActionClassName-ActionName-validation.xml
- 系统寻找到第一个校验文件时还会继续搜索后面的校验文件，当搜索到所有校验文件时，会把校验文件里的所有校验规则汇总，然后全部应用于action方法的校验。如果两个校验文件中指定的校验规则冲突，则只使用后面文件中的校验规则。

- 当action继承了另一个action，父类action的校验文件会先被搜索到。

- 假设UserAction继承BaseAction：
- <action name="user" class="cn.itcast.action.UserAction" method="{1}">
- </action>

- 访问上面action，系统先搜索父类的校验文件：BaseAction-validation.xml，BaseAction-user-validation.xml，接着搜索子类的校验文件： UserAction-validation.xml， UserAction-user-validation.xml。应用于上面action的校验规则为这四个文件的总和