动态ActionForm与验证框架

动态ActionForm

- 完整类名是org.apache.struts.action.DynaActionForm
- 最重要的功能就是用配置的方式来代替编程
- 用strtus框架的时候需要配置大量的ActionForm传输数据，如果项目很大，那么ActionForm类的数量就会很多，而且需要提供大量的get和set方法，而使用DynaActionForm，则无需创建任何类，一切都在配置文件中完成

配置ActionForm

<form-bean name="自定义名称"
type="org.apache.struts.action.DynaActionForm">

<form-property name="属性名" property="属性数据类型">
<form-property name="属性名" property="属性数据类型">
……
</form-bean>

- 动态表单用<form-property>配置ActionForm的属性，有几个属性就设置几个<form-property>
- 如果属性的类型是基本类型，必须用包装类类型，例如java.lang.Integer代表int类型

配置动态ActionForm示例

```
<form-bean name="myDynaForm"
    type="org.apache.struts.action.DynaActionForm">
      <form-property name="username" type="java.lang.String" />
      <form-property name="prodname" type="java.lang.String" />
      <form-property name="qty" type="java.lang.String" />
      <form-property name="phone" type="java.lang.String" />
</form-bean>
```

在Action中使用动态ActionForm

- 使用动态ActionForm时，请求参数必须使用DynaActionForm的 getter方法获取。
- DynaActionForm的getter方法主要有如下三个。
  - String getString(String name)：根据属性名返回字符串类型的属性值
  - Object get(String name)：根据属性名返回对象类型的值。
  - Object get(String name , int index)：对于有多个重名表单域的情况， Struts将其当成数组处理，此处根据表面域名和索引获取对应值。

在Action中使用动态ActionForm示例

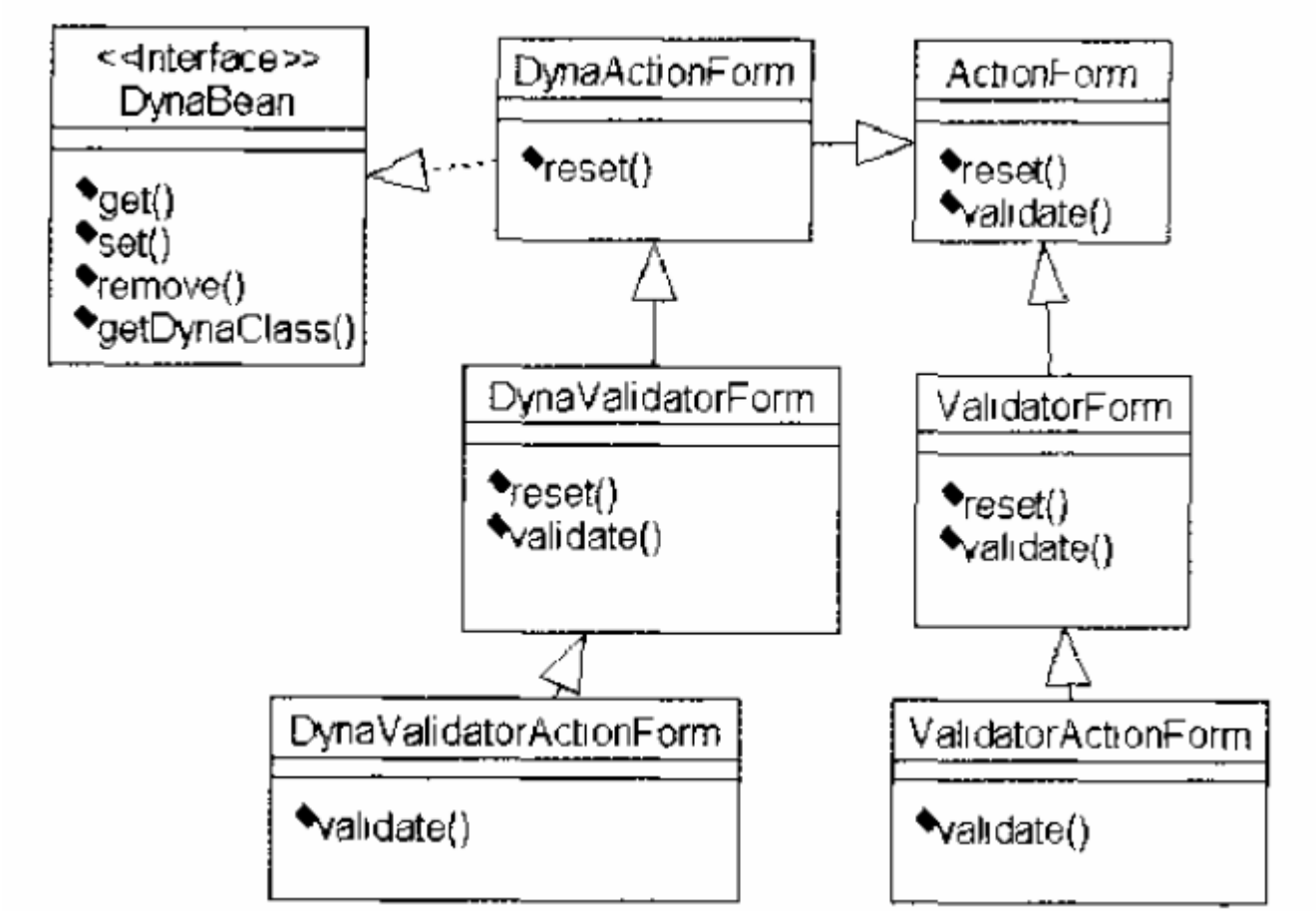
```
DynaActionForm myform = (DynaActionForm) form;

String username = myform.getString("username");
String prodname = myform.getString("prodname");
String qty = myform.getString("qty");
String phone = myform.getString("phone");
```

动态ActionForm与普通ActionForm

- 使用动态ActionForm 的目的是为了减少代码的书写量。但有些IDE 工具可以自动生成ActionForm.则可以使用普通ActionForm。

- 动态ActionForm与普通ActionForm并没有太大的区别。动态ActionForm避免了书写ActionForm，但配置变得更复杂了。而普通ActionForm使解析请求参数变得更直观。

ActionForm的关系图



Validate框架验证

- Struts两种验证方式：
  - 1:ActionForm中validate方法验证
  - 2:Validate框架验证
- ActionForm验证
  - 很多表单都要验证输入不能为空，传统方法需要在每个ActionForm中写if语句，相当于同样的语句分散在项目的很多地方
- Validate框架验证
  - 将验证的工作写在一个XML文件中,同样的语句只需要写一次就好了

使用Validate框架之前的准备

- jar文件
  - jakarta-oro.jar
  - commons-validator.jar
- xml文件
  - validator-rules.xml 通用规则验证，一般没有必要修改
  - validation.xml 针对于具体某个验证，需要具体自定义配置

Validate框架验证实现

- 1:建立validator-rules.xml和 validation.xml
- 2:将这两个文件作为插件存入struts配置文件，在struts配置文件最后加上：

```
<plug-in className="org.apache.struts.validator.ValidatorPlugIn">
    <set-property property="pathnames" value="/WEB-INF/validator-rules.xml,/WEB-INF/validation.xml" />
</plug-in>
```

validation.xml文件标签

```
<form-validation>
    <global>
        <constant>
            <constant-name>xxx</constant-name>
            <constant-value>xxx</constant-value>
        </constant>
    </global>
    <formset>
        <form name="xxx">
                        <field property="xxx" depends="xxx,xxx">
                                <msg name="xxx" key="xxx"/>
                                <msg name="xxx" key="xxx"/>

                                <var>
                                        <var-name>xxx</var-name>
                                        <var-value>xxx</var-value>
                                </var>
                        </field>
            </form>
     </formset>
</form-validation>
```

<form-validation> 元素

- 文件根元素，包含两个子元素
  - <global>
  - <formset>

<global>元素

- 可以定义<contant>子元素，定义常量表达式，在文件的其他地方用${xxx}形式引用
- 例如：

```
<global>
    <constant>
        <constant-name>postalCode</constant-name>
        <constant-value>^\d{5}\d*$</constant-value>
    </constant>
</global>
```

<formset>元素

- 可以包含一个至多个<form>子元素，用来配置多个表单验证

<form>元素

- <form>元素对应一个表单的验证
- 属性：
  - name 指定了ActionForm的名字
- 子元素
  - 可以 包含多个<field>子元素

<field>元素

- <field>元素对应表单的一个属性字段
- 属性
  - property 需要验证字段的名字
  - depends 验证的规则，多个规则用逗号隔开
- 子元素
  - <msg> 对应的消息文本，将替代默认的消息文本
  - <var> 向验证规则传递参数
  - <arg0><arg1><arg2><arg3>给消息文本传递参数

<msg>和<var>

- <msg> 设置当验证失败时指定的错误消息
  - 属性
    - name 指定验证规则
    - key 指定资源文件错误消息
    - resource 如果为true，则认为使用资源文件的key，如果为false，则表示直接使用key指定的字符串，默认为true
- <var> 设置验证规则的参数，例如指定正则表达式，例如设置最大值或最小值
  - 子元素
    - <var-name> 规则名称
    - <var-value> 指定值

<arg>参数

- <arg0>到<arg3>可以设置消息文本的{n}参数
- 属性同msg
  - name 指定验证规则，省略则表示适应所有的规则
  - key 指定资源文件错误消息
  - resource 如果为true，则认为使用资源文件的key，如果为false，则表示直接使用key指定的字符串，默认为true

验证框架针对的ActionForm

- validate验证框架不能针对于标准的ActionForm进行验证
- 如果是标准的ActionForm
  - ActionForm应该继承org.apache.struts.validator.ValidatorForm
- 如果是动态ActionForm
  - 动态ActionForm类型应该是 org.apache.struts.validator.DynaValidatorForm
- 不管是哪种ActionForm，对于验证框架的配置都是一样的

validator-rules.xml自带的通用验证规则

- validator-rules.xml包含了一些通用验证规则，对所有struts应用 都适用，我们可以在validation.xml文件中直接调用以下验证规则
  - required 不能为空
  - validwhen 条件验证
  - minlength 最小长度
  - maxlength 最大长度
  - mask 正则表达式
  - double 是否是double
  - integer 是否是整数
  - date 是否是日期
  - intRange 整数范围
  - doubleRange double范围
  - email 是否是email

required验证

- 强制某个域必须填写，不能为空
- 示例：

```
<field property="t1" depends="required">
        <msg name="required" key="test.error.msg"/>
</field>
```

minlength验证

- 最小的字符长度，需要传递一个参数设置最小长度值
- 示例：

```
<field property="t1" depends="minlength">
    <msg name="minlength" key="test.error.msg"/>
    <var>
        <var-name>minlength</var-name>
        <var-value>3</var-value>
    </var>
</field>
```

maxlength验证

- 最多的字符长度，需要传入一个参数设置最大字符数
- 示例：

```
<field property="t1" depends="maxlength">
    <msg name="maxlength" key="test.error.msg"/>
    <var>
        <var-name>maxlength</var-name>
        <var-value>3</var-value>
    </var>
</field>
```

mask验证

- 通过一个正则表达式来验证输入域的值是否满足正则表达式，需要传入一个参数设置正则表达式
- 示例：

```
<field property="t1" depends="mask">
    <msg name="mask" key="test.error.msg"/>
    <var>
        <var-name>mask</var-name>
        <var-value>^\d{3,8}$</var-value>
    </var>
</field>
```

integer验证

- 是否是一个整数数字
- 示例：

```
<field property="t1" depends="integer">
    <msg name="integer" key="test.error.msg"/>
</field>
```

date验证

- 是否是一个正确格式的日期时间
- 示例：

```
<field property="t1" depends="date">
    <msg name="date" key="test.error.msg"/>
    <var>
        <var-name>datePattern</var-name>
        <var-value>yyyy-MM-dd</var-value>
    </var>
</field>
```

intRange验证

- 判断输入的值是否在一个整数范围之内，需要传入最小值和最大值，并且依赖于integer验证
- 示例：

```
<field property="t1" depends="integer,intRange">
    <msg name="integer" key="test.error.msg"/>
    <msg name="intRange" key="test.error.msg"/>
    <var>
        <var-name>min</var-name>
        <var-value>18</var-value>
    </var>
    <var>
        <var-name>max</var-name>
        <var-value>60</var-value>
    </var>
</field>
```

验证密码是否相同

- 可以用validwhen设置test表达式来验证密码是否相同
- 示例：

```
<field property="t1" depends="required">
    <msg name="required" key="test.error.msg"/>
</field>
<field property="t2" depends="validwhen">
    <msg name="validwhen" key="test.error.msg"/>
    <var>
        <var-name>test</var-name>
        <var-value>(*this* == t1)</var-value>
    </var>
</field>
```

添加javascript验证

- 在JSP页面中加入

  <html:javascript formName=“ActionForm名称"/>
- 在页面中会自动生成javascript验证函数
- 在<html:form>标签中加入

  onsubmit=“return 验证函数名(this)”
- 当用户提交时会自动调用javascript进行客户端验证