JAVA开发应用学习笔记

郝俊禹*

Contents

*Email:haojunyu2012@gmail.com

1 Struts2

1.1 开发环境的搭建

1.1.1 所需的jar包

struts-core-2.x.x.jar Struts2框架的核心类库

xwork-2.x.x.jar XWork类库,Struts 2在其上构建

ognl-2.6.x.jar 对象图导航语言,struts2框架通过其读写对象的属性

freemarker-2.3.x.jar Struts2的UI标签的模板使用FreeMarker编写

commons-logging-1.1.x.jar ASF出品的日志包,Struts2框架使用这个日志包来支持Log4j和JDK1.4+的日志记录

commons-fileupload-1.2.1.jar 文件上传组件,2.1.6版本后必须加入此文件

1.1.2 编写配置文件struts.xml

```
1  <?xml version="1.0" encoding="UTF-8" ?>
2  <!DOCTYPE struts PUBLIC "-//Apache Software Foundation//DTD
       Struts Configuration 2.1//EN" "http://struts.apache.org/
       dtds/struts-2.1.dtd">
3  <struts>
4      <package name="myactions" namespace="/"
5          extends="struts-default">
6          <action name="login" class="com.cm.LoginAction">
7              <result name="success">/success.jsp</result>
8              <result name="error">/error.jsp</result>
9          </action>
10     </package>
11 </struts>
```

- package属性说明

  name 与java里面的包功能类似,将相同功能的action放在一个包里,易于管理

  namespace namespace作为action中访问路径path属性的一部分,可以简化代码

  extend 默认继承struts-default包–Struts2很多核心的功能都是拦截器来实现的, 不继承struts-default就无法使用struts2提供的核心功能

- action属性说明

  name action名称,也可作为action路径的一部分.可以用通配符*,用来执行不同的方法

  class 处理的类,默认的是ActionSupport

  method 默认的是execute方法来执行

  result 视图部分,用来导向x.jsp文件

param 通过struts.xml传递参数给action类(用set方法接受),然后进行处理

```
<struts>
    <package name="myactions" namespace="/"
        extends="struts-default">
        <action name="login_*" class="com.cm.LoginAction"
            method="{1}">
            <result name="success">/success.jsp</result>
        </action>
    </package>
</struts>
```

若访问的action名为login_addUI,这会匹配该action,并执行addUI方法.

- 常用常量

struts.action.extension 设定访问action文件时用的后缀名

```
<struts>
    <constant name="struts.action.extension" value="
        do,action"/>
</struts>
```

struts.i18n.encoding 指定默认编码集,作用于HttpServletRequest的setCharacterEncoding和freemarker,velovity的输出

```
<struts>
    <constant name="struts.i18n.encoding" value="UTF
        -8"/>
</struts>
```

struts.configuration.xml.reload 当struts.xml文件修改后,系统自动重新加载改文件, 默认值为false,开发阶段最好打开

```
<struts>
    <constant name="struts.configuration.xml.reload"
        value="true"/>
</struts>
```

struts.enable.DynamicMethodInvocation 设置struts2是否允许使用动态方法调用

```
<struts>
    <constant name="struts.enable.
        DynamicMethodInvocation" value="false"/>
</struts>
```

struts.multipart.maxSize 设置struts2能上传文件的大小限制

```
<struts>
    <constant name="struts.multipart.maxSize" value="
        10701096"/>
</struts>
```

- 多个配置文件
  当系统中action数量大量增加后,会导致struts.xml配置文件变得非常臃肿.为提高struts.xml文件的可读性,可以将一个struts.xml配置文件分解成多个配置文件,然后在struts.xml中包含其他配置文件即可.

```
<struts>
    <include file="struts-user.xml"/>
    <include file="struts-order.xml"/>
</struts>
```

1.1.3 添加StrutsMVC框架启动配置

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
____http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">

  <display-name></display-name>

  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>

  <filter>
    <filter-name>struts2</filter-name>
    <filter-class>
        org.apache.struts2.dispatcher.ng.filter.
            StrutsPrepareAndExecuteFilter
    </filter-class>
  </filter>

  <filter-mapping>
    <filter-name>struts2</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping></web-app>
```

注:自从Sturts2.1.3以后,org.apache.struts2.FilterDispatcher已经标注为过时了.

1.2 struts2的处理流程

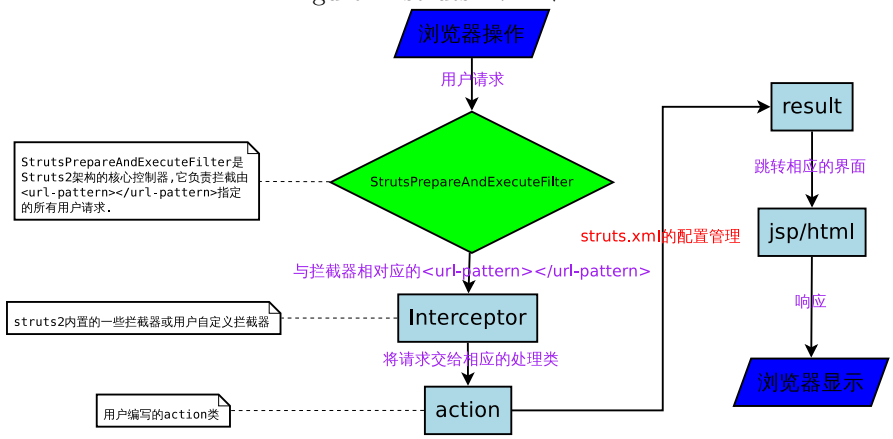
流程如下图1所示:

1.3 struts2中的session/application/request

1.3.1 一般用法

```
public String execute() {
    //usage——only to access the attributes of the three
        objects
    ActionContext ctx = ActionContext.getContext();
```

Figure 1: struts2的工作流程




```
4       ctx.getApplication().put("app", "应用范
            围");      //put app into ServletContext
5       ctx.getSession().put("ses", "范
            围session"); //put ses into session
6       ctx.put("req", "范
            围request");                  //put req into request
7       ctx.put("names", Arrays.asList("老王","老张","老徐"));
8       return "mySession";
9   }
```

1.3.2 另一种用法

```
1   //access it directly by class——ServletActionContext
2   public String rsa() throws Exception{
3       //usage——get the real path(servletContext.getRealPath) of
            the website
4       HttpServletRequest request = ServletActionContext.
            getRequest();
5       ServletContext servletContext = ServletActionContext.
            getServletContext();
6       request.setAttribute("req", "请求范围属性");
7       request.getSession().setAttribute("ses", "会话范围属性");
8       servletContext.setAttribute("app", "应用范围属性");
9       //HttpServletResponse response = ServletActionContext.
            getResponse();
10      return "mySession";
11  }
```

1.4 文件上传

1.4.1 所需的jar包

在WEB-INF/lib下加入下面的jar包

1. commons-fileupload-1.2.1.jar
2. commons-io-1.3.2.jar

1.4.2 设置form表的enctype

将form表的enctype设置为"multipart/form-data"

```
//access it directly by class——ServletActionContext
public String rsa() throws Exception{
    //usage——get the real path(servletContext.getRealPath) of the website
    HttpServletRequest request = ServletActionContext.getRequest();
    ServletContext servletContext = ServletActionContext.getServletContext();
    request.setAttribute("req", "请求范围属性");
    request.getSession().setAttribute("ses", "会话范围属性");
    servletContext.setAttribute("app", "应用范围属性");
    //HttpServletResponse response = ServletActionContext.getResponse();
    return "mySession";
}
```

1.5 struts2中的拦截器

1.5.1 类型转换器

- 局部类型转换器只对某个action起作用

1. 自定义类型转换器

```
package com.hjy.type.converter;

import java.sql.Date;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Map;

import com.opensymphony.xwork2.conversion.impl.DefaultTypeConverter;

public class DateTypeConverter extends DefaultTypeConverter {

    @Override
    public Object convertValue(Map<String, Object> context, Object value,
            Class toType) {
        // TODO Auto-generated method stub
        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyyMMdd");
        try{
```

```
18              if(toType == Date.class){
                        //String—Date
19                  String[] params = (String[]) value;
                            //requese.getParameterValues()
                        —all parameters
20                  return dateFormat.parse(params[0]);
21              }else if(toType ==String.class){
                        //Date—String
22                  Date date = (Date) value;
23                  return dateFormat.format(date);
24              }
25          }catch(ParseException e){
26              return null;
27          }
28          return super.convertValue(context, value,
                toType);
29      }
30  }
```

2. 注册局部类型转换器在Action类所在的包下放置ActionClassName-conversion.properties文件, ActionClassName是Action的类名,后面的-conversion.properties是固定写法.在properties文件中的内容为**属性名称=类型转换器的全类名**.

- 全局类型转换器对整个应用都起作用

1. 自定义类型转换器
   方法见局部类型转换器.
2. 注册全局类型转换器在src所在的包下放置xwork-conversion.properties文件,后面的-conversion.properties是固定写法.在properties文件中的内容为**待转换的类型=类型转换器的全类名**.

1.5.2 自定义权限拦截器

- 该类实现的接口
  com.opensymphony.xwork2.interceptor.Interceptor
- 在intercept方法中添加拦截的操作

```
1  @Override
2  public String intercept(ActionInvocation invocation)
       throws Exception {
3      // TODO Auto-generated method stub
4      Object user = ActionContext.getContext().getSession()
           .get("user");
5
6      if(user != null)              //user != null means the
           user has logged in,who is allow to execute the
           methods
7          return invocation.invoke();
8      ActionContext.getContext().put("message","你没有权限执行
           改操作");
9      return "message";
10 }
```

- 在struts.xml配置拦截器并在action中使用

```
<interceptors>
    <interceptor name="permission" class="com.hjy.
        interceptor.PermissionInterceptor"></interceptor>
    <interceptor-stack name="permissionStack">
        <interceptor-ref name="defaultStack"></
            interceptor-ref>
        <interceptor-ref name="permission"></interceptor-
            ref>
    </interceptor-stack>
</interceptors>

<action name="login" class="com.hjy.action.LoginAction">
    <interceptor-ref name="permissionStack"></interceptor
        -ref>
</action>
```

1.5.3 输入校验

两种方法:

- 采用手工编写代码实现
  - 继承actionsupport类 com.opensymphony.xwork2.ActionSupport;
  - 重写validate()方法–针对action中所有方法

```
@Override
public void validate() {            //validate all
    the methods in the action
    // TODO Auto-generated method stub
    if(this.username == null || this.username.trim().
        equals("")){
        this.addFieldError("username", "用户名不为空");
    }

    if(this.password == null || this.password.trim().
        equals("")){
        this.addFieldError("password", "密码不为空");
    }else{
        if(this.password.length() < 4){
            this.addFieldError("password", "密码长度少
                与6");
        }
    }
}
```

  - 重写validateXxx()方法–针对action中xxx方法

```
@Override
public void validateUpdate() {          //validate
    all the methods in the action
    // TODO Auto-generated method stub
```

```
4       if(this.username == null || this.username.trim().
            equals("")){
5           this.addFieldError("username", "用户名不为空");
6       }
7
8       if(this.password == null || this.password.trim().
            equals("")){
9           this.addFieldError("password", "密码不为空");
10      }else{
11          if(this.password.length() < 4){
12              this.addFieldError("password", "密码长度少
                    与6");
13          }
14      }
15  }
```

– 验证失败后,请求转发给input视图

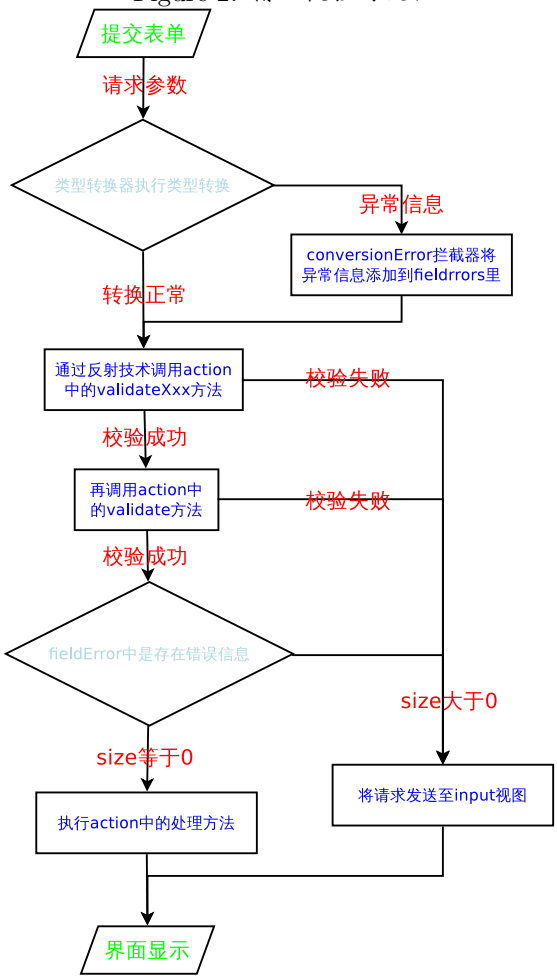
```
1 <result name="input">/WEB-INF/login.jsp</result>
```

– 显示
到input视图的界面用<s:fielderror>显示失败信息
输入校验的流程图2如下所示:

- 基于XML配置方式实现
  – 继承actionsupport类
  com.opensymphony.xwork2.ActionSupport;
  – 新建校验文件,格式为ActionClassName-validation.xml,对所有action中方法有效的模板如下

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE validators PUBLIC
3      "-//OpenSymphony Group//XWork Validator 1.0.3//EN
           "
4      "http://www.opensymphony.com/xwork/xwork-
           validator-1.0.3.dtd">
5
6
7  <validators>
8      <field name="username">
9          <field-validator type="requiredstring">
10             <param name="trim">true</param>
11             <message用户名不能为空>!</message>
12         </field-validator>
13     </field>
14     <field name="password">
15         <field-validator type="requiredstring">
16             <message密码不能为空>!</message>
17         </field-validator>
18
19         <field-validator type="stringlength">指定
20             <!--属性最小长度user-->
```

Figure 2: 输入校验的流程



```
21                  <param name="minLength">4</param>
22                  <message 密码长度要大于>4!</message>
23              </field-validator>
24          </field>
25  </validators>
```

– 新建校验文件,格式为ActionClassName-actionName-validation.xml,对某个action方法有效的模板如下

– 校验器种类

required 必填校验器,要求filed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.

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,改逻辑表达式基于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.

– 验证失败后,请求转发给input视图

```
1 <result name="input">/WEB-INF/login.jsp</result>
```

– 显示
到input视图的界面用<s:fielderror>显示失败信息

1.6 FAQ

1.6.1 接受中文请求参数乱码问题

出现这种情况,一般用户使用的是struts2.1.6版本或之前的,原因是struts2.1.6在获取并使用了请求参数后才调用HttpServletRequest的setCharacterEncoding()方

法进行编码设置,导致应用使用的就是乱码请求参数,这个bug 在struts2.1.8中已经被解决.

解决方法如下: 新建一个Filter,把这个Filter放置在struts2的Filter之前,然后在doFilter()方法里添加以下代码:

```
public void doFilter(...){
    HttpServletRequest req = (HttpServletRequest)request;
    req.setCharacterEncoding("UTF-8");   //replace UTF-8 with
        the encode you use
    filterchain.doFilter(request,response);
}
```