Java 企业级开发(J2EE)实验报告

指导教师：王磊

课程名称：JavaEE 技术	年级：xxx	实验日期：
姓名：xxx	学号：xxxxxxx	班级：xxx
实验名称：Struts2 框架编程	实验序号：实验四	成员人数：1

一、 实验目的及要求

实验目的：

1、使用 OGNL 进行参数传递；
2、熟悉 Struts2 的拦截器，Struts2 中的许多特性都是通过拦截器来实现的；
3、Struts 可以对用户输入提供一些基本的合法性验证。

实验原理：

1、所有的请求发给前端控制器；
2、前端控制器维护一个配置文件(配置文件中指明了不同的请求和某个 Action 的对应关系)；
3、前端控制器可以根据请求的不同，调用不同的 Action；
4、Action 调用 Model(模型层:实体或数据访问)，实现业务功能，数据放在 request 中；
5、请求转发给 View 层（JSP）
6、JSP 显示数据

二、实验环境(仪器与材料)

Windows 7/10，MySQL/MariaDB，Eclipse,Tomcat

三、实验内容及完成情况

1、新建 Java Web 项目 actionMethod，在 lib 文件夹中引入 jar 包。如图 1 所示。

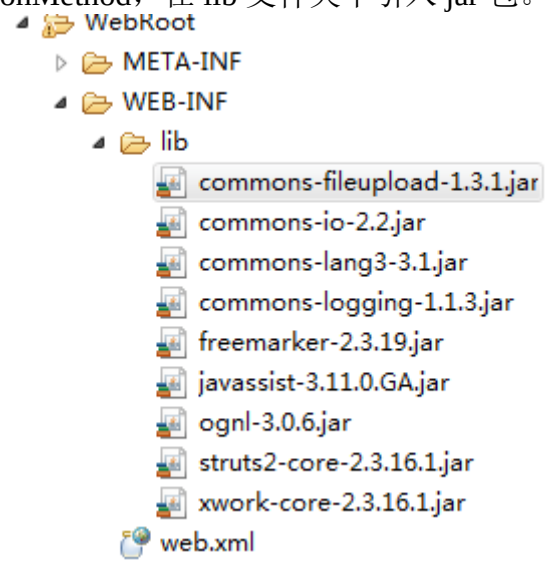

图1 在lib文件夹下加入jar包

WebRoot下jsp页面如图2所示。注意将全体jsp页面编码方式设置为utf-8。

图2 项目中的jsp页面

2、在src目录下创建包com.bbc.model和com.bbc.action，并分别创建类。

在包model下面创建User类，代码如下：

```
package com.bbc.model;
public class User {
	private String username;
	private String password;
	public String getUsername() {
		return username;
	}
	public void setUsername(String username) {
		this.username = username;
	}
	public String getPassword() {
		return password;
	}
	public void setPassword(String password) {
		this.password = password;
	}
}
```

在包action下面创建UserAction类，代码如下：

```
package com.bbc.action;
import java.util.ArrayList;
import java.util.List;
import com.bbc.model.User;
import com.opensymphony.xwork2.ActionContext;
import com.opensymphony.xwork2.ActionSupport;
public class UserAction extends ActionSupport {
	private User user;
	private List<String> list=new ArrayList<String>();
	public List<String> getList() {
		return list;
	}
	public void setList(List<String> list) {
		this.list = list;
	}
	public User getUser() {
		return user;
	}
	public void setUser(User user) {
		this.user = user;
```

```
	}
		@Override
	public String execute() throws Exception {
		return SUCCESS;
	}
	public String login(){
		if(user.getUsername().equals("admin")&& user.getPassword().equals("123")){
			list.add("user1");
			list.add("user2");
			ActionContext.getContext().put("list",list);
			return SUCCESS;
		}else{
			return INPUT;
		}
	}
	public String add(){
		return SUCCESS;
	}
	public String delete(){
		return SUCCESS;
	}
	public String update(){
		return "update";
	}
	public String query(){
		return "query";
	}
	public void validateLogin() {
		if(user.getUsername()==null||user.getUsername().equals("")){
			this.addFieldError("username", "用户名不能为空");
		}
		if(user.getPassword()==null||user.getPassword().equals("")){
			this.addFieldError("password", "密码不能为空");
		}
	}
}
```

3、在src目录下创建struts.xml。

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC "-//Apache Software Foundation//DTD Struts Configuration 2.1//EN"
"http://struts.apache.org/dtds/struts-2.1.dtd">
<struts>
<constant name="struts.enable.DynamicMethodInvocation" value="true" />
<package name="struts2" namespace="/" extends="struts-default">
	<action name="userlogin" class="com.bbc.action.UserAction" method="login">
		<result name="success">/main.jsp</result>
		<result name="input">/login.jsp</result>
	</action>
	<action name="useradd" class="com.bbc.action.UserAction" method="add">
		<result name="success">/useradd.jsp</result>
	</action>
	<action name="userdelete" class="com.bbc.action.UserAction" method="delete">
		<result name="success">/userdelete.jsp</result>
	</action>
```

```
        <action name="user" class="com.bbc.action.UserAction">
              <result name="update">/userupdate.jsp</result>
              <result name="query">/userquery.jsp</result>
        </action>
</package>
</struts>
```

注意：使用DMI动态调用方式，红色的语句必须添加。

4、打开web.xml配置文件

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

红色字体部分为新添加内容

5、编写login.jsp页面，运行界面如图3所示。

```
<body>
   <center>
   <form action="userlogin" method="post">
        用户名：<input type="text" name="user.username"/><br>
        密　码：<input type="password" name="user.password"/><br>
        <input type="submit" value="提交"/>
    </form>
    <s:fielderror></s:fielderror>
    </center>
  </body>
```

图3 系统启动运行界面

6、编写main.jsp页面。Login.jsp中输入admin和123，运行结果如图4所示。

```
<%@ taglib prefix ="s" uri="/struts-tags"%>
<body>
  <s:debug></s:debug>
  <center>
     <h1>用户登录成功</h1>
  当前登录用户名：<s:property value='User.username'/> </br></br></br>
    <a href="useradd">添加用户</a>
```

```
    <a href="userdelete">删除用户</a>
    <a href="user!update">修改用户</a>
    <a href="user!query">查询用户</a>
    </center>
    </br>
    显示在action中存入  Value Stack Contents中的list,直接使用s标签和 OGNL 表达式即可</br>
    <s:property value="list[0]"/>
    <s:property value="list[1]"/>
  <hr>
  显示在action中存入Stack Context 中的list，必须在OGNL表达式前加 # </br>
    <s:property value="#list[0]"/>
    <s:property value="#list[1]"/>
  </body>
```

[Debug]

图4 用户登录成功界面

7、编写添加用户、删除用户界面，main.jsp使用方法调用（method）

(1)useradd.jsp

```
<body>
    <center>添加用户成功</center>
</body>
```

(2)userdelete.jsp

```
<body>
    <center>用户删除成功</center>
</body>
```

main.jsp中发出的请求语句如下：

```
<a href="useradd">添加用户</a>
<a href="userdelete">删除用户</a>
```

8、编写查询用户、修改用户界面，main.jsp使用DMI动态调用方式，可以接收！后面的参数。

(1)userquery.jsp

```
<body>
<center>用户查询页面</center>
</body>
```

(2)userupdate.jsp

```
<body>
<center>用户修改页面</center>
</body>
```

main.jsp中发出的请求语句如下：

```
<a href="user!update">修改用户</a>
<a href="user!query">查询用户</a>
```

9、在UserAction中添加验证代码

```
public void validateLogin() {
        if(user.getUsername()==null||user.getUsername().equals("")){
                this.addFieldError("username", "用户名不能为空");
        }
        if(user.getPassword()==null||user.getPassword().equals("")){
                this.addFieldError("password", "密码不能为空");
```

```
    }
}
```

并在login.jsp中添加代码<s:fielderror></s:fielderror>，测试效果如图5-7。

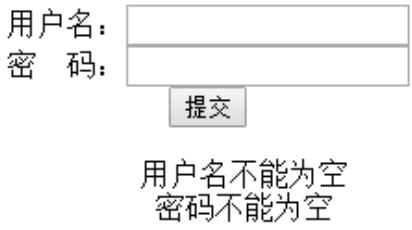

图5 测试均为空情况

图6 测试密码为空情况

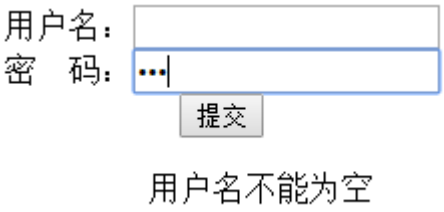

图7 测试用户名为空情况

10、Debug界面

点击main.jsp页面的Debug链接，进入Debug界面，如图8所示。可以在该界面下观察ValueStatck（栈）和Stack Context(映射)两种数据结构。

注意：Statck栈结构中数据，可以使用s标签和OGNL表达式直接取出，而map映射结构中，必须在s标签OGNL表达式前面加上#。在main.jsp页面代码中均有体现。

[Debug]

Struts ValueStack Debug

Value Stack Contents

Object	Property Name	Property Value
com.bbc.action.UserAction	texts	null
	actionErrors	[]
	errors	{}
	fieldErrors	{}
	errorMessages	[]
	container	There is no read method for container
	locale	zh_CN
	actionMessages	[]
	list	[user1, user2]
	user	com.bbc.model.User@7a9df636
com.opensymphony.xwork2.DefaultTextProvider	texts	null

Stack Context

These items are available using the #key notation

Key	Value
com.opensymphony.xwork2.dispatcher.HttpServletRequest	org.apache.struts2.dispatcher.StrutsRequestWrapper@7de627ca
com.opensymphony.xwork2.ActionContext.locale	zh_CN
com.opensymphony.xwork2.dispatcher.HttpServletResponse	org.apache.catalina.connector.ResponseFacade@6b5ab921
com.opensymphony.xwork2.ActionContext.name	userlogin
	{org.apache.jasper.compiler.TldLocationsCache=org.apache.jasper.compiler.TldLoca org.apache.tomcat.InstanceManager=org.apache.catalina.core.DefaultInstanceManage Files/Apache Software Foundation/Tomcat 7.0/webapps/actionmethod/WEB-INF/classes

图 8 Debug 界面

四、 出现的问题及解决方案

Struts2 的 action 具体视图的返回可以由用户自己定义的 Action 来决定,根据返回的字符串找到对应的配置项，来决定视图的内容。具体 Action 的实现可以是一个普通的 Java 类，里面有

publicString execute 方法即可，或者实现 Action 接口，不过最常用的是从 ActionSupport 继承，好处在于可以直接使用 Struts2 封装好的方法。

五、教师评语