Java 应用框架 Struts

实验指导书

软件学院

实验一、熟悉 struts 环境、工具

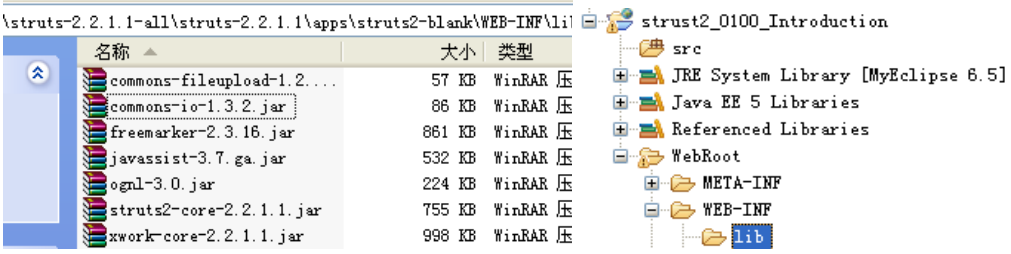
实验目的：通过完成 HelloWorld 程序熟悉 Struts2 的开发流程和工具

实验类型：验证

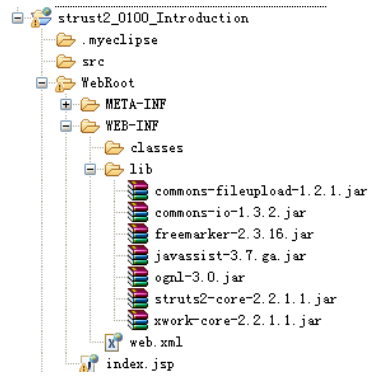
实验要求：必做

实验内容：

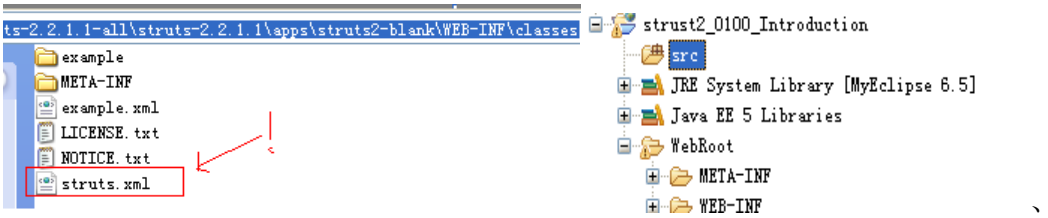
- 1 在 MyEclipse 中新建 web 工程
- 2 在 struts-2.2.1.1-all\struts-2.2.1.1 解压 struts2-blank.war(最基础的示例程序)
- 3 拷类库，在这个项目的 lib 文件下面拷



把 jar 放入 lib 后看不见 jar 文件，是因为 MyEclipse 默认视图是 package Explorer，如果要看硬盘上对应的视图，应该打开 windows-Show View-other-navigatior



- 4 进入 struts-2.2.1.1\apps\struts2-blank\WEB-INF\classes 下把 struts.xml 拷到 web 工程的 src 下面，因为工程编译完它默认就把 src 下的文件放到 class 文件下面。



- 5 配置 web.xml,参考 struts 自带的 web.xml，把 filter 的配置拷过来

```
<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>
```

示例程序 HelloWorld

✧ 在 webroot 目录下编写 3 个 jsp 页面，index.jsp、Success.jsp 和 Error.jsp。

- index.jsp

```
<%@ page language="java" import="java.util.*" pageEncoding="utf-8"%>
<%@ taglib prefix=”s” uri=”/struts-tags” %>
<%
    String path = request.getContextPath();
    String basePath = request.getScheme() + "://"
            + request.getServerName() + ":" + request.getServerPort()
            + path + "/";
%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
    <head>
        <base href="<%=basePath%>">
        <title>My JSP 'index.jsp' starting page</title>
    </head>

    <body>
        <!-- 提交到action  -->
        <form action="login.action" method="post">
            用户名:
             <!-- 参数名和action中的属性名一样  -->
            <input type=text name=username>
            <br>
            密  码:
            <input type=password name=userpass>
            <br>
            <input type=submit name=subm value="提交">
            <input type=reset name=reset value="取消">
        </form>
    </body>
</html>
```

- Success.jsp

```
<%@ page language="java" import="java.util.*" pageEncoding="utf-8"%>
<%
    String path = request.getContextPath();
    String basePath = request.getScheme() + "://"
            + request.getServerName() + ":" + request.getServerPort()
            + path + "/";
%>
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
    <head>
        <base href="<%=basePath%>">
        <title>My JSP 'Success.jsp' starting page</title>
    </head>
    <body>
        <h1>
            欢迎
            <%=request.getParameter("username")%>
            ,登录
        </h1>
    </body>
</html>
```

- Error.jsp

```
<%@ page language="java" import="java.util.*" pageEncoding="utf-8"%>
<%
    String path = request.getContextPath();
    String basePath = request.getScheme() + "://"
            + request.getServerName() + ":" + request.getServerPort()
            + path + "/";
%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
    <head>
        <base href="<%=basePath%>">
        <title>My JSP 'Error.jsp' starting page</title>
    </head>
    <body>
        <h1>
            用户名或密码错误!
        </h1>
    </body>
</html>
```

✧ 在 src 目录下建立一个类，包名 mypack，类名 UserAction，其代码如下：

```
package mypack;
import com.opensymphony.xwork2.ActionSupport;
public class UserAction extends ActionSupport {
    private String username;
    private String userpass;
    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
```

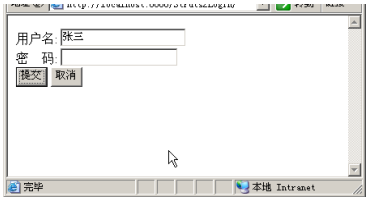
```
    }
    public String getUserpass() {
        return userpass;
    }
    public void setUserpass(String userpass) {
        this.userpass = userpass;
    }
    @Override
    public String execute() throws Exception {
        if ("Mike".equals(username) && "123".equals(userpass)
                || "张三".equals(username) && "abc".equals(userpass))
            return "success";
        else
            return "error";
    }
}
```

✧ 在 src 目录下建立 Struts2 的配置文件 struts.xml，内容如下：

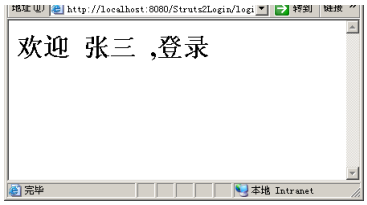
```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
   "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
   "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
    <!-- 定义包管理配置的action 继承struts-default.xml中的配置 -->
    <package name="actions" extends="struts-default">
    <!-- 定义Action(login.action)  -->
        <action name="login" class="mypack.UserAction">
            <!-- 定义转发路径对应的字符串名  -->
            <result name="success">/Success.jsp</result>
            <result name="error">/Error.jsp</result>
        </action>
    </package>
</struts>
```

✧ 程序部署和运行。

http://localhost:8080/helloworld/



index.jsp



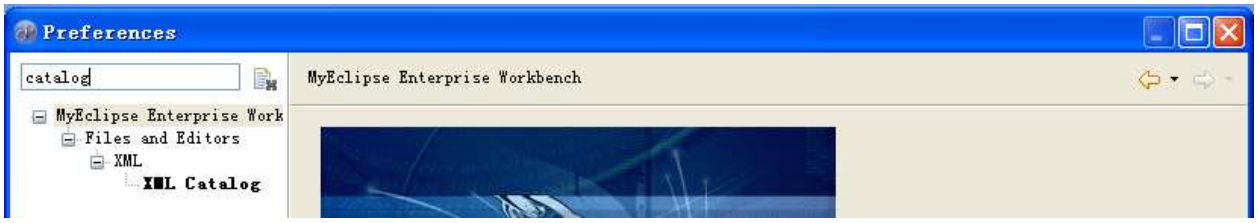
Success.jsp

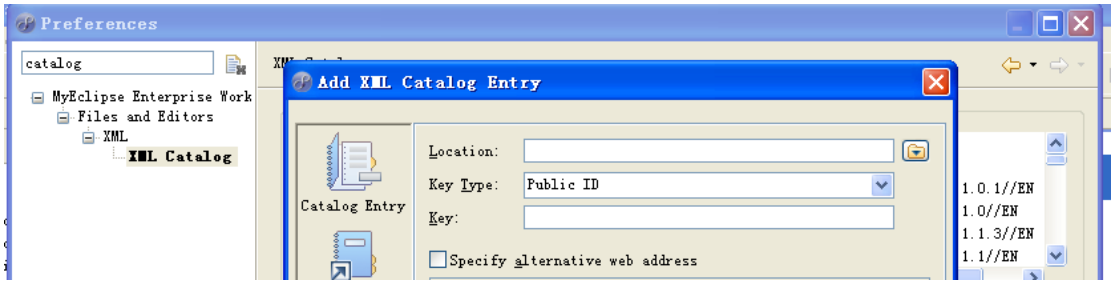
敲尖括号不提示的问题

Struts.xml 文件头定义了

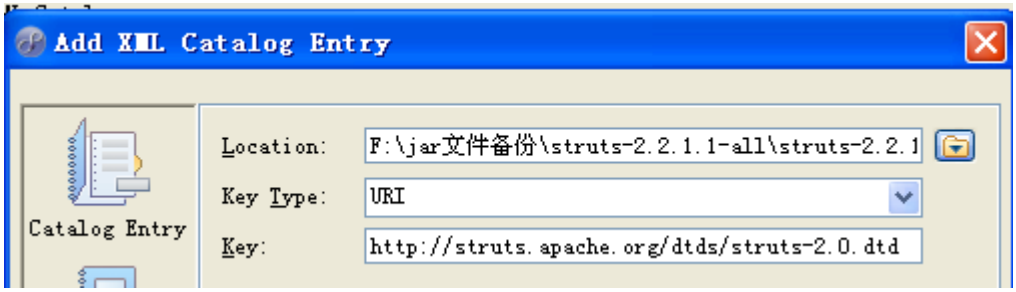
```
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
    "http://struts.apache.org/dtds/struts-2.0.dtd">
```

配置：windows---preferences---catalog---。。。xml—xml Catalog—Add





Add 入本地定义当前 xml 的 dtd 文件：找到 struts2-core-2.2.1.1.jar 解压开找到 struts-2.1.7.dtd



完成，验证代码提示成功！

Struts的namespace

示例工程Struts2_0200_Namespace

Struts.xml

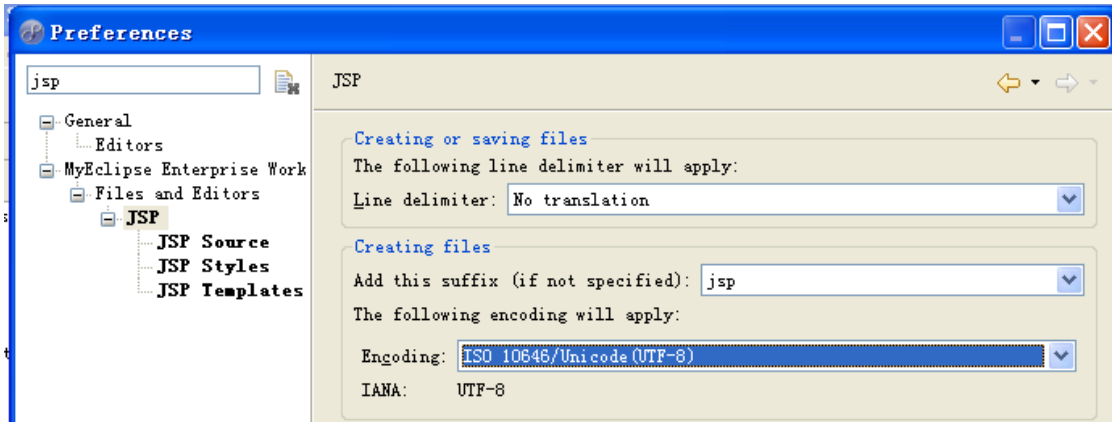
```
<struts>
    <constant name="struts.devMode" value="true" />
    <package name="front" extends="struts-default" namespace="/front">
        <action name="index">
            <result>/Namespace.jsp</result>
        </action>
    </package>
     <package name="main" extends="struts-default" namespace="">
        <action name="index">
            <result>/Namespace.jsp</result>
        </action>
    </package>
</struts>
```

所以 namespace 为空意味着：只要找到一个 index.action，没有找到精确的对应的 namespace，全部都交给 namespace 为空的这个 package 去处理，所以这个 package 囊括了其他所有 package 处理不了的 action。

Struts 自定义具体视图的返回

示例工程 Struts2_0300_Action

- 修改 jsp 模板字符编码：windows-preferences- JSP 修改编码为 UTF-8 即可



IndexAction1.java

```
public class IndexAction1 {
    public String execute() {
        return "success";
    }
}
```

IndexAction2.java

```
public class IndexAction2 implements Action {
```

```
    public String execute() {
        return "success";
    }
}
```

真正企业开发只用这第三种！另外两种忘记！

IndexAction3.java

```
public class IndexAction3 extends ActionSupport {
    public String execute() {
        return "success";
    }
}
```

```
<struts>
    <constant name="struts.devMode" value="true" />
    <package name="front" extends="struts-default" namespace="/">
        <action name="index" class="com.bjsxt.struts2.front.action.IndexAction1">
            <result name="success">/ActionIntroduction.jsp</result>
        </action>
    </package>
</struts>
```

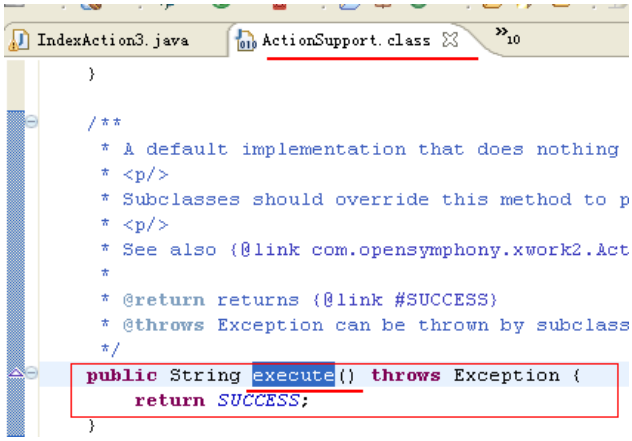
具体视图的返回可以由用户自己定义的 Action 来决定

具体的手段是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 public String 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



如果不配置 class 属性，默认执行 xwork 框架的 ActionSupport 这个 action，这个 action 就有 execute 这个方法，return success。

实验二、Struts2 基础

实验目的：掌握 Struts2 中 Action 的不同的实现方法

实验类型：验证

实验要求：必做

实验内容：

- Action 有三种接收参数的方式

1. 属性接收参数
2. 用 DomainModel（实体模型）接收参数
3. 用 ModelDriven 接收参数（ 不常用 ）

■ **用Action的属性接收参数**

Index.jsp

```
<head>
<base href="<%=basePath %>"/></head>
使用action属性接收参数<a href="userAdd.action?name=a&age=8">添加用户</a>
</body>
```

怎么接受参数的呢？第一种方式.在自己的 action 下面定义两个属性，写好 get,set 方法，当 new 完 action 的时候，会自动把这两个属性从参数里面拿过来，帮你设置好。

参数跟我们的成员变量一一对应，这时候它就会自动把我们的参数传递到我们成员变量里。这时候当我们调用 add()方法时，它直接可以用了。

UserAction.java

```
public class UserAction extends ActionSupport {
    private String name;
    private int age;
    public String execute() {
        System.out.println("name=" + name);
        System.out.println("age=" + age);
        return SUCCESS;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
```

```
}
```

Struts.xml

```
<struts>
    <package name="user" extends="struts-default" namespace="/user">
 <action name="userAdd" class="com.bjsxt.struts2.user.action.UserAction">
          <result>/user_add_success.jsp</result>
       </action>
    </package>
</struts>
```

■ **使用Domain Model (实体模型) 接收参数**

```
<html>
<body> 使用Domain Model接收参数
<a href="user/user.action?user.name=a&user.age=8">添加用户</a>
</body>
</html>
public class UserAction extends ActionSupport {
    private User user;
    //private UserDTO userDTO;
    public String execute() {
        System.out.println("name=" + user.getName());
        System.out.println("age=" + user.getAge());
        return SUCCESS;
    }
    public User getUser() {
        return user;
    }
    public void setUser(User user) {
        this.user = user;
    }
}
public class User {
    private String name;
    private int age;
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
```

}

- **多方法的 Action**

```
package action;
import po.Userinfo;
import service.UserService;
import com.opensymphony.xwork2.ActionContext;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.ModelDriven;
public class CrudUserAction extends ActionSupport implements
ModelDriven<Userinfo> {
    // crud 业务方法
    private UserService userservice = new UserService();
     private  Userinfo userinfo=new Userinfo();
    // 模型对象 userinfo
    public Userinfo getModel() {
        // TODO Auto-generated method stub
        return userinfo;
    }
    // 增加
    public String create() throws Exception {
        userservice.createUser(userinfo);
        return SUCCESS;
    }
    // 查询
    public String retrive() throws Exception {
        // 查询结果放在 request 中
    ActionContext.getContext().put("userlist", userservice.selectUsers());
    return "list";
    }
    // 修改
    public String update() throws Exception {
        userservice.updateUser(userinfo);
        return SUCCESS;
    }
    // 删除
    public String delete() throws Exception {
        userservice.deleteUser(userinfo.getUsername());
        return SUCCESS;
    }
    // 默认的 execute
    public String execute() throws Exception {
        return SUCCESS;
    }
}
```

在 struts.xml 中配置如下：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
   "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
   "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
    <package name="actions" extends="struts-default">
        <action name="CrudUser" class="action.CrudUserAction">
            <result>/Success.jsp</result>
          <result name="list">/UserList.jsp</result>
        </action>
    </package>
</struts>
```

实验三、Struts2 的转发控制

实验目的：掌握 Struts2 中不同 Result 类型配置方法及区别

实验类型：验证

实验要求：必做

实验内容：

```
<struts>
    <package name="resultTypes" namespace="/" extends="struts-default">
        <action name="r1">
        <result type="dispatcher">/r1.jsp</result>
        </action>
        <action name="r2">
        <result type="redirect">/r2.jsp</result>
        </action>
        <action name="r3">
        <result type="chain">r1</result>
        </action>
        <action name="r4">
        <result type="redirectAction">r2</result>
        </action>
    </package>
</struts>
```

Result 类型

1. dispatcher
2. redirect
3. chain
4. redirectAction
5. freemarker
6. httpheader
7. stream
8. velocity
9. xslt
10. plaintext
11. tiles

实验四、Struts2 的输入校验

实验目的：掌握 Struts2 中 validate 方法来进行验证

实验类型：验证

实验要求：必做

实验内容：

validate 方法验证

```
package action;
import javax.servlet.http.HttpServletRequest;
import org.apache.struts2.ServletActionContext;
import oper.UserOper;
import po.Userinfo;
import com.opensymphony.xwork2.ActionContext;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.ModelDriven;
public class UserAction extends ActionSupport implements ModelDriven<Userinfo>
{
    // 数据模型
    private Userinfo user = new Userinfo();
    // 业务类
    private UserOper uo = new UserOper();
    public Userinfo getModel() {
        // TODO Auto-generated method stub
        return user;
    }
    // 增加前
    public String precreate() throws Exception {
        return "addupdate";
    }
    // 增加
    public String create() throws Exception {
        uo.create(user);
        return select();
    }
    // 删除
    public String delete() throws Exception {
        uo.delete(user.getUserid());
        return select();
    }
    // 修改
    public String update() throws Exception {
        uo.update(user);
```

```
    return select();
}
// 查询
public String select() throws Exception {
    ActionContext.getContext().put("userlist", uo.retriveAll());
    return SUCCESS;
}
// 查询单个
public String retrive() throws Exception {
    Userinfo myuser = uo.retriveOne(user.getUserid());
    user.setUsername(myuser.getUsername());
    user.setUserpass(myuser.getUserpass());
    user.setSex(myuser.getSex());
    user.setSfz(myuser.getSfz());
    user.setBirthday(myuser.getBirthday());
    user.setWorktime(myuser.getWorktime());
    user.setEmail(myuser.getEmail());
    user.setInterest(myuser.getInterest());
    user.setIntro(myuser.getIntro());
    user.setXl(myuser.getXl());
    return "addupdate";
}
// 验证方法
// 增加时验证
public void validateCreate() {
    checkForm();
    // 用户名是否存在
    if(uo.checkUserName(user.getUsername()))
        this.addFieldError("username", "用户名已经被占用");
}
// 修改时验证
public void validateUpdate() {
    checkForm();
    if(uo.checkUserName(user.getUsername(),user.getUserid()))
        this.addFieldError("username", "用户名已经被占用");
}
// 验证要求
public void checkForm() {
    HttpServletRequest request = ServletActionContext.getRequest();
    // 用户名不能为空
    if (user.getUsername() == null || user.getUsername().equals(""))
        this.addFieldError("username", "用户名不能为空");
    // 密码不能为空
    if (user.getUserpass() == null || user.getUsername().equals(""))
        this.addFieldError("userpass", "密码不能为空");
```

```
        // 2次密码相同
        else if
(!user.getUserpass().equals(request.getParameter("userpass1")))
            this.addFieldError("userpass", "2次密码不一样");
        // 身份证不能为空
        if (user.getSfz() == null || user.getSfz().equals(""))
            this.addFieldError("sfz", "身份证不能为空");
        // 身份证必须是15或18位
        else if (!user.getSfz().matches("^\\d{17}[\\d|X]|\\d{15}$"))
            this.addFieldError("sfz", "身份证必须是15位或18位");
        // email不能为空
        if (user.getEmail() == null || user.getEmail().equals(""))
            this.addFieldError("email", "Email不能为空");
        else if (!user.getEmail().matches("^\\w+@\\w+(\\.\\w+)+$"))
            this.addFieldError("email", "Email格式错误");
        // 生日不能为空
        if (user.getBirthday() == null)
            this.addFieldError("birthday", "生日不能为空");
        // 兴趣至少选一项
        if (user.getInterest() == null)
            this.addFieldError("interest", "兴趣至少选一项");
        // 工作年限1-100之间
        if (user.getWorktime() < 1 || user.getWorktime() > 100)
            this.addFieldError("worktime", "工作年限必须在1-100之间");
        // 简介不能为空
        if (user.getIntro() == null || user.getIntro().equals(""))
            this.addFieldError("intro", "简介不能为空");
    }
}
```

- Struts.xml 中增加了验证出错时跳转到的页面<result name=”input”></result>。

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
    "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
    <!-- 请求参数的中文处理 -->
    <constant name="struts.i18n.encoding" value="GBK"/>
    <!-- 修改后的xml自动加载 -->
    <constant name="struts.configuration.xml.reload" value="true"/>
     <package name="actions" extends="struts-default">
        <action name="user" class="action.UserAction">
          <result>/UserList.jsp</result>
          <result name="addupdate">/AddUpdate.jsp</result>
          <result name="input">/AddUpdate.jsp</result>
        </action>
```

```
    </package>
</struts>
```

validate 框架(xml)验证

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE validators PUBLIC
       "-//OpenSymphony Group//XWork Validator 1.0.2//EN"
       "http://www.opensymphony.com/xwork/xwork-validator-1.0.2.dtd">
<!-- 验证规则定义根元素 -->
<validators>
    <!-- 第一个验证字段：姓名name -->
    <field name="name">
        <!-- 验证规则：非空(系统预先定义好的规则) -->
        <field-validator type="requiredstring">
            <!-- 错误提示 -->
            <message>姓名不能为空</message>
        </field-validator>
        <!-- 验证规则：长度在5-10之间(系统预先定义好的规则) -->
        <field-validator type="stringlength">
            <param name="minLength">5</param>
            <param name="maxLength">10</param>
            <!-- 错误提示 -->
            <message>姓名长度必须在${minLength}-${maxLength}个字符之间</message>
        </field-validator>
    </field>
    <!-- 第二个验证字段：年龄age -->
    <field name="age">
        <field-validator type="conversion" short-circuit="true">
            <!-- 错误提示 -->
            <message>必须输入整数</message>
       </field-validator>
        <field-validator type="int">
            <param name="min">20</param>
            <param name="max">35</param>
            <!-- 错误提示 -->
            <message>年龄必须在${min}-${max}之间</message>
        </field-validator>
    </field>
    <!-- 第三个验证字段：分数mark -->
    <field name="mark">
       <field-validator type="conversion" short-circuit="true">
            <!-- 错误提示 -->
            <message>必须输入数字</message>
        </field-validator>
        <field-validator type="double">
```

```
            <param name="minInclusive">50</param>
            <param name="maxInclusive">100</param>
            <!-- 错误提示 -->
            <message>分数必须在${minInclusive}-${maxInclusive}之间</message>
        </field-validator>
    </field>
    <!-- 第三个验证字段：入学时间enrolltime -->
    <field name="enrolltime">
    <field-validator type="conversion" short-circuit="true">
            <!-- 错误提示 -->
            <message>必须是日期格式</message>
        </field-validator>
        <field-validator type="required">
            <!-- 错误提示 -->
            <message>入学时间不能为空</message>
        </field-validator>
        <field-validator type="date">
            <param name="min">1990-01-01</param>
            <param name="max">2008-10-09</param>
            <!-- 错误提示 -->
            <message>入学时间必须在${min}到${max}之间</message>
        </field-validator>
    </field>
</validators>
```

实验五、Struts2 的标签（一）

实验目的：熟悉 Struts 标签库

实验类型：验证

实验要求：必做

实验内容：

- if~elseif~else

描述：执行基本的条件流转。

名称	必需	默认	类型	描述	备注
test	是		boolean	决定标志里的内容是否显示的表达式	else 标志没有这个参数
id	否		Object/String	用来标识元素的 id。在 UI 和表单中为 HTML 的 id 属性	

例：

```
<s:set name="age" value="61"/>
<s:if test="${age > 60}">
    老年人
</s:if>
<s:elseif test="${age > 35}">
    中年人
</s:elseif>
<s:elseif test="${age > 15}" id="wawa">
    青年人
</s:elseif>
<s:else>
    少年
</s:else>
```

- Iterator（迭代）

描述：用于遍历集合（java.util.Collection）或枚举值(java.util.iterator)

名称	必需	默认	类型	描述
status	否		String	如果设置此参数，一个 IteratorStatus 的实例将会压入每一个遍历的堆栈
value	否		Object/ String	要遍历的可枚举的(iteratable)数据源，或者将放入的新列表(List)的对想
id	否		Object/ String	用来标识元素的 id。在 ui 和表单中为 HTML 的 id 属性

- I18n（国际化操作）

描述：加载资源包到值堆栈。它可以允许 text 标志访问任何资源包的信息。而不只当前的 action 相关联的资源包。

名称	必需	默认	类型	描述
name	是		Object/String	资源包的类路径(如 com.xxxx.resources.AppMsg)
id	否		Object/String	用来标识元素的 id。在 ui 和表单中为 HTML 的 id 属性

- Include

描述：包含一个 servlet 的输出(servlet 或 jsp 的页面)

名称	必需	默认	类型	描述
value	时		String	要包含的 jsp 页面或 servlet
id	否		Object/String	用来标识元素的 id。在 ui 和表单中为 HTML 的 id 属性

- param

描述：属性是可选的，如果提供，会调用 Component 的方法，addParameter(String,Object),如果不提供，则外层嵌套标签必须实现 UnnamedParametric 接口。

Value 的提供有两种方式，通过 value 属性或者标签中间的 text，不同之处：

<s:param name="name">zhaosoft</s:param>

参数会以 String 的格式放入 statck

<s:param name="name" value="zhaosoft"/>

该值会以 java.lang.Object 的格式放入 statck

名称	必需	默认	类型	描述
name	否		String	参数名
value	时		String	value 表达式
id	否		Object/String	用来标识元素的 id。在 ui 和表单中为 HTML 的 id 属性

- set

描述：set 标签赋予变量一个特定范围内的值。当希望给一个变量赋一个复杂的表达式，每次访问该变量而不是复杂的表达式时用到。其在两种情况下非常有用：复杂的表达式很耗时（性能提升）或者很难理解（代码的可读性提高）

名称	必需	默认	类型	描述

name	是		String	变量名字
scope	否		String	变量作用域，可以为 application,session,request,page,action
value	否		Object/String	将会赋给变量的值
id	否		Object/String	用来标识元素的 id。在 ui 和表单中为 HTML 的 id 属性

- Text

描述：支持国际化信息的标签。国际化信息必须放在一个和当前 action 同名的 resource bundle 中，如果没有找到相应 message,tag body 将被当作默认的 message，如果没有 tag body，message 的 name 会被作为默认 message.

名称	必需	默认	类型	描述
name	是		String	资源属性的名字
id	否		Object/String	用来标识元素的 id。在 ui 和表单中为 HTML 的 id 属性

- url

描述：该标签用于创建 url，可以通过”param”标签提供 request 参数。

当 includeParams 的值是 all 或 get，param 标签中定义的参数将有优先权，也就是说其会覆盖其他同名参数的值。

实验六、Struts2 的标签（二）

实验目的：熟悉 Struts 标签库

实验类型：设计

实验要求：必做

实验内容：

- **单行文本框**

Textfield 标签输出一个 HTML 单行文本输入控件，等价于 HTML 代码<input type="text">

名称	必需	默认	类型	描述
maxlength	否	无	Integer	文本输入控件可以输入字符的最大长度
readonly	否	false	Boolean	当该属性为 true 时，不能输入
size	否	无	Integer	指定可视尺寸
id	否		Object/String	用来标识元素的 id。在 ui 和表单中为 HTML 的 id 属性

例：

```
<s:form action="register" method="post">
   <s:textfield name="username" label="用户名"></s:textfield>
</s:form>
```

- **文本框区**

Textarea 标签输出一个 HTML 多行文本输入控件，等价于 HTML 代码：〈textarea /〉

名称	必需	默认	类型	描述
cols	否	无	Integer	列数
rows	否	无	Integer	行数
readonly	否	false	Boolean	当该属性为 true 时，不能输入
wrap	否	false	Boolean	指定多行文本输入控件是否应该换行
id	否		Object/String	用来标识元素的 id。在 ui 和表单中为 HTML 的 id 属性

例：

```
<s:textarea name="personal" cols="10" rows="5" label="个人简历"></s:textarea>
```

- **下拉列表**

s:select 标签输出一个下拉列表框，相当于 HTML 代码中的<select/>

名称	必需	默认	类型	描述
list	是	无	Cellection Map Enumeration Iterator array	要迭代的集合，使用集合中的元素来设置各个选项，如果 list 的属性为 Map 则 Map 的 key 成为选项的 value，Map 的 value 会成为选项的内容
listKey	否	无	String	指定集合对象中的哪个属性作为选项的 value

listValue	否	无	String	指定集合对象中的哪个属性作为选项的内容
headerKey	否	无	String	设置当用户选择了 header 选项时，提交的的 value，如果使用该属性，不能为该属性设置空值
headerValue	否	无	String	显示在页面中 header 选项内容
emptyOption	否	false	Boolean	是否在 header 选项后面添加一个空选项
multiple	否	false	Boolean	是否多选
size	否	无	Integer	显示的选项个数

例：

```
<%@ page contentType="text/html;charset=GBK" %>
<%@ taglib prefix="s" uri="/struts-tags" %>

<h3>使用name和list属性，list属性的值是一个列表</h3>
<s:form>
    <s:select label="最高学历" name="education" list="{'高中','大学','硕士','博士'}"/>
</s:form>

<h3>使用name和list属性，list属性的值是一个Map</h3>
<s:form>
    <s:select label="最高学历" name="education" list="#{1:'高中',2:'大学',3:'硕士',4:'博士
'}"/>
</s:form>

<h3>使用headerKey和headerValue属性设置header选项</h3>
<s:form>
    <s:select label="最高学历" name="education" list="{'高中','大学','硕士','博士'}"
       headerKey="-1" headerValue="请选择您的学历"/>
</s:form>


<h3>使用emptyOption属性在header选项后添加一个空的选项</h3>
<s:form>
    <s:select label="最高学历" name="education" list="{'高中','大学','硕士','博士'}"
       headerKey="-1" headerValue="请选择您的学历"
       emptyOption="true"/>
</s:form>


<h3>使用multiple属性设置多选</h3>
<s:form>
    <s:select label="最高学历" name="education" list="{'高中','大学','硕士','博士'}"
       headerKey="-1" headerValue="请选择您的学历"
       emptyOption="true"
       multiple="true"/>
```

```
</s:form>

<h3>使用size属性设置下拉框可显示的选项个数</h3>
<s:form>
    <s:select label="最高学历" name="education" list="{'高中','大学','硕士','博士'}"
       headerKey="-1" headerValue="请选择您的学历"
       emptyOption="true"
       multiple="true" size="8"/>
</s:form>


<h3>使用listKey和listValue属性，利用Action实例的属性（property）来设置选项的值和选项的内容</h3>
<s:form>
    <s:select label="最高学历" name="education" list="educations"
       listKey="id" listValue="name"/>
</s:form
```

- **doubleselect 标签**

doubleselect 标签输出关联的两个 HTML 列表框，产生联动效果。

名称	必需	默认	类型	描述
list	是	无	Cellection Map Enumeration Iterator array	要迭代的集合，使用集合中的元素来设置各个选项，如果 list 的属性为 Map 则 Map 的 key 成为选项的 value，Map 的 value 会成为选项的内容
listKey	否	无	String	指定集合对象中的哪个属性作为选项的 value，该选项只对第一个列表框起作用
listValue	否	无	String	指定集合对象中的哪个属性作为选项的内容，该选项只对第一个列表框起作用
headerKey	否	无	String	设置当用户选择了 header 选项时，提交的的 value，如果使用该属性，不能为该属性设置空值
headerValue	否	无	String	显示在页面中 header 选项内容
emptyOption	否	false	Boolean	
multiple	否	false	Boolean	是否多选
size	否	无	Integer	显示的选项个数，该选项只对第一个列表框起作用
doubleId	否	无	String	指定第二个列表框的 ID
doubleList	是	无	Cellection Map Enumeration Iterator array	要迭代的集合

doubleListKey	否	无	String	指定集合对象中的哪个属性作为选项的 value，该选项只对第二个列表框起作用
doubleListValue	否	无	String	指定集合对象中的哪个属性作为选项的内容，该选项只对第二个列表框起作用
doubleSize	否	无	Integer	选项个数
doubleName	否	无	String	指定第二个列表框的 name 映射
doubleValue	否	无	Object	第二个列表框的初始选种项

例：

```
<s:form name="test">
    <s:doubleselect label="请选择所在省市"
       name="province" list="{'四川省','山东省'}" doubleName="city"
       doubleList="top == '四川省' ? {'成都市', '绵阳市'} : {'济南市', '青岛市'}" />
</s:form>

<s:form action="doubleselectTag">
    <s:doubleselect
            label="请选择所在省市"
            name="province"
            list="provinces"
            listKey="id"
            listValue="name"
            doubleList="cities"
            doubleListKey="id"
            doubleListValue="name"
            doubleName="city"
            headerKey="-1"
            headerValue="---------- 请选择 ----------"
            emptyOption="true" />
</s:form>
```

- 复选框

名称	必需	默认	类型	描述
fieldValue	是	true	String	指定在复选框选中时，实际提交的值

- 复选框组，对应 Action 中的集合

名称	必需	默认	类型	描述
list	是	无	Collection Map Enumeration Iterator array	要迭代的集合，使用集合中的元素来设置各个选项，如果 list 的属性为 Map 则 Map 的 key 成为选项的 value，Map 的 value 会成为选项的内容
listKey	否	无	String	指定集合对象中的哪个属性作为选项的 value

listValue	否	无	String	指定集合对象中的哪个属性作为选项的内容

例：checkboxlistTag.jsp

```
<%@ page contentType="text/html;charset=GBK" %>
<%@ taglib prefix="s" uri="/struts-tags" %>
<s:form>
    <s:checkboxlist name="interest" list="{'足球','篮球','排球','游泳'}" label="兴趣爱好"/>
</s:form>
```

- **单击复选框**

名称	必需	默认	类型	描述
accept	否	input	String	HTML accept 属性，指出接受文件的 MIME 类型

- **按钮**

Submit 标签输出一个按钮，submit 标签和 form 标签使用可以提供异步表单提交功能。Submit 标签可以输出以下三种类型的提交按钮：

Input： 等价于 HTML 代码<input type=" submit" >

Image： 等价于 HTML 代码 <input type="image">

Button：<button type="submit"></button>

名称	必需	默认	类型	描述
type	否	input	String	要使用的提交按钮的类型，有效值：input button image
src	否	无	String	为 image 类型的提交按钮设置图片地址。该属性对 input 和 button 类型的提交按钮无效
action	否	无	String	指定处理提交请求的 action
method	否	无	String	指定处理提交请求的 action 的方法

例1:

```
<s:submit type="image" method="login" src="images/login.jpg"></s:submit>
```

页面输出：

```
<input type="image" alt="Submit" src="images/login.jpg" id="user__login"
name="method:login" value="Submit"/>
```

例2:

```
<s:submit type="button" action="selectTag" method="login" label="登陆"></s:submit>
```

页面输出：

```
<button type="submit" id="user_selectTag_login" name="action:selectTag!login"
value="Submit">登陆</button>
```

Struts2 预定义的前缀：

1、method前缀

使用method前缀，来取代action默认的execute()方法的执行。

```
<s:form action="user">
```

```
    <s:textfield name="user.username" label="用户名"></s:textfield>
    <s:textfield name="user.password" label="密码"></s:textfield>
    <s:submit value="登陆" name="method:login"></s:submit>
    <s:submit value="注册" name="method:register"></s:submit>
</s:form>
```

注意：input类型的按钮，不能用label设置按钮上的文本，只能用value

2、action前缀

使用action前缀，取代form标签指定的action，导向到另一个action进行处理。

```
<s:form action="login">
    <s:textfield name="user.username" label="用户名"></s:textfield>
    <s:textfield name="user.password" label="密码"></s:textfield>
    <s:submit value="登陆"></s:submit>
    <s:submit value="注册" name="action:register"></s:submit>
</s:form>
```

3、redirect前缀

使用redirect前缀请求重定向到其他的url，甚至可以是web英语程序外部的url。

```
<s:form action="login">
    <s:textfield name="user.username" label="用户名"></s:textfield>
    <s:textfield name="user.password" label="密码"></s:textfield>
    <s:submit value="登陆"></s:submit>
    <s:submit value="搜索" name="redirect:www.google.com"></s:submit>
</s:form>
```

- **reset 标签**

reset标签输出一个重置按钮

名称	必需	默认	类型	描述
type	否	input	String	要使用的重置按钮的内容，input、button

```
<s:reset value="重置"></s:reset>
<s:reset type="button" label="重置"></s:reset>
```

- **updownselect 标签**

updownselect标签创建一个带有上下移动的按钮的列表框，可以通过上下移动按钮来调整列表框的选项的位置。

名称	必需	默认	类型	描述
list	是	无	Cellection Map Enumeration Iterator array	要迭代的集合，使用集合中的元素来设置各个选项，如果 list 的属性为 Map 则 Map 的 key 成为选项的 value，Map 的 value 会成为选项的内容
listKey	否	无	String	指定集合对象中的哪个属性作为选项的 value
listValue	否	无	String	指定集合对象中的哪个属性作为选项的内容

headerKey	否	无	String	设置当用户选择了 header 选项时，提交的的 value，如果使用该属性，不能为该属性设置空值
headerValue	否	无	String	显示在页面中 header 选项内容
emptyOption	否	false	Boolean	是否在 header 选项后面添加一个空选项
multiple	否	false	Boolean	是否多选
size	否	无	Integer	显示的选项个数
moveUplabel	否		String	设置向上移动按钮上的文本
moveDownLabel	否		String	设置向下移动按钮上的文本
selectAllLabel	否		String	设置向全部选择按钮上的文本
allowMoveUp	否	无	Boolean	设置是否使用向上移动按钮
allowMoveDown	否	无	Boolean	设置是否使用向下移动按钮
allowSelectAll	否	无	Boolean	设置是否使用全部选择按钮

实例：

```
<s:form>
<!-- 使用简单集合来生成可上下移动选项的下拉选择框 -->
<s:updownselect name="a" label="请选择您喜欢的图书" labelposition="top"
    moveUpLabel="向上移动"
    list="{'Spring2.0宝典' , '轻量级J2EE企业应用实战' , 'JavaScript: The Definitive Guide'}"/>


<!-- 使用简单Map对象来生成可上下移动选项的下拉选择框
     且使用emptyOption="true"增加一个空选项-->
<s:updownselect name="b" label="请选择您想选择出版日期" labelposition="top"
    moveDownLabel="向下移动"
    list="#{'Spring2.0宝典':'2006年10月' , '轻量级J2EE企业应用实战':'2007月4月' , '基于J2EE的
Ajax宝典':'2007年6月'}"
    listKey="key"
    emptyOption="true"
    listValue="value"/>

<s:bean name="com.zhaosoft.ui.formtag.BookService" id="bs"/>
<!-- 使用集合里放多个JavaBean实例来可上下移动选项的生成下拉选择框 -->
<s:updownselect name="c" label="请选择您喜欢的图书的作者" labelposition="top"
    selectAllLabel="全部选择" multiple="true"
    list="#bs.books"
    listKey="author"
    listValue="name"/>
</s:form>

package com.zhaosoft.ui.formtag;
```

```
public class BookService
{
    public Book[] getBooks()
    {
       return new Book[]
       {
           new Book("Spring2.0宝典","zhaosoft"),
           new Book("轻量级J2EE企业应用实战","zhaosoft"),
           new Book("基于J2EE的Ajax宝典","zhaosoft")
       };
    }
}
```

- optiontransferselect 标签

optiontransferselect标签创建一个选项转移列表组建，它由两个<select>标签以及它们之间的用于将选项在两个<select>之间相互移动的按钮组成。表单提交时，将提交两个列表框中选中的选项。

名称	必需	默认	类型	描述
list	是	无	Cellection Map Enumeration Iterator array	要迭代的集合，使用集合中的元素来设置各个选项，如果 list 的属性为 Map 则 Map 的 key 成为选项的 value，Map 的 value 会成为选项的内容，该选项只对第一个列表框起作用
listKey	否	无	String	指定集合对象中的哪个属性作为选项的 value，该选项只对第一个列表框起作用
listValue	否	无	String	指定集合对象中的哪个属性作为选项的内容，该选项只对第一个列表框起作用
headerKey	否	无	String	设置当用户选择了 header 选项时，提交的的 value，如果使用该属性，不能为该属性设置空值
headerValue	否	无	String	显示在页面中 header 选项内容
multiple	否	false	Boolean	是否多选
size	否	无	Integer	显示的选项个数，该选项只对第一个列表框起作用
doubleId	否	无	String	指定第二个列表框的 ID
doubleList	是	无	Cellection Map Enumeration Iterator array	要迭代的集合，使用集合中的元素来设置各个选项，如果 doubleList 的属性为 Map 则 Map 的 key 成为选项的 value，Map 的 value 会成为选项的内容，该选项只对第二个列表框起作用

doubleListKey	否	无	String	指定集合对象中的哪个属性作为选项的 value，该选项只对第二个列表框起作用
doubleListValue	否	无	String	指定集合对象中的哪个属性作为选项的内容，该选项只对第二个列表框起作用
doubleHeaderKey	否	无	String	设置当用户选择了 header 选项时，提交的的 value，如果使用该属性，不能为该属性设置空值
doubleHeaderVale	否	无	String	显示在页面中 header 选项内容
doubleEmptyOption	否	无	String	是否在第二列表框的 header 后面添加一个空选项
doubleMultiple	否	true	Boolean	是否多选
doubleSize	否	无	Integer	选项个数
doubleName	否	无	String	指定第二个列表框的 name 映射
doubleValue	否	无	Object	第二个列表框的初始选种项
leftTitle	否	无	String	左边列表框的标题
rightTitle	否	<-	String	右边列表框的标题
addToLeftLable	否		String	
addToRightLable				
addAllToLeftLable				
addAllToRightLable				
leftUpLabel				
leftDownLabel				
rightUpLabel				
rightDownLabel				
allowAddToLeft				
allowAddToright				
allowAddAllToLeft				
allowAddAllToRight				
allowSelectAll	否	无	Boolean	设置是否使用全部选择按钮
allowUpdownOnLeft				
allowUpDownOnRight				

例:

```
<s:head/>
<s:form>
    <s:optiontransferselect
```

```
    label="最喜爱的图书"
        name="javaBook"
        list="{'《Java Web开发详解》', '《Struts 2深入详解》', '《Java快速入门》'}"
        doubleName="cBook"
        doubleList="{'《VC++深入详解》', '《C++ Primer》', '《C++程序设计语言》'}"/>
</s:form>
--------------------------
<s:form>
    <s:optiontransferselect
    label="最喜爱的图书"
        name="book1"
        leftTitle="Java图书"
        rightTitle="C/C++图书"
        list="{'《Java Web开发详解》', '《Struts 2深入详解》', '《Java快速入门》'}"
        headerKey="-1"
        headerValue="--- 请选择 ---"
        emptyOption="true"
        doubleName="book2"
        doubleList="{'《VC++深入详解》', '《C++ Primer》', '《C++程序设计语言》'}"
        doubleHeaderKey="-1"
        doubleHeaderValue="--- 请选择 ---"
        doubleEmptyOption="true"
        addToLeftLabel="向左移动"
        addToRightLabel="向右移动"
        addAllToLeftLabel="全部左移"
        addAllToRightLabel="全部右移"
        selectAllLabel="全部选择"
        leftUpLabel="向上移动"
        leftDownLabel="向下移动"
        rightUpLabel="向上移动"
        rightDownLabel="向下移动"/>
</s:form>
```

实验七、Struts2 的国际化

实验目的：掌握 Struts2 的国际化方法

实验类型：设计

实验要求：必做

实验内容：

- 在 struts.xml 中配置 struts.custom.i18n.resources 常量

<constant name="struts.custom.i18n.resources" value="globalMessages"/>

- 在 src 目录下建立中文和英文的资源文件，中文资源文件 globalMessages_zh_CN.properties 的内容如下：

username=用户名
userpass=密码
success=登录成功
error=登录失败
login=登录

使用 native2ascii 工具把该文件转换为 unicode 编码。

- 英文资源文件 globalMessages_en.properties 的内容如下：

username=User Name
userpass=Password
success=Welcome
error=Sorry! You can not log in
login=Login

- 编写登录页面 Login.jsp，其内容如下：

```
<%@ page language="java" import="java.util.*" pageEncoding="utf-8"%>
<%@ taglib uri="/struts-tags" prefix="s" %>
<%
    String path = request.getContextPath();
    String basePath = request.getScheme() + "://"
            + request.getServerName() + ":" + request.getServerPort()
            + path + "/";
%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
    <head>
        <base href="<%=basePath%>">
        <title>My JSP 'Login.jsp' starting page</title>
    </head>
    <body>
    <s:form action="dologin">
    <s:textfield name="username" key="username"/>
    <s:password name="userpass" key="userpass"/>
    <s:submit key="login"/>
```

```
    </s:form>
    </body>
</html>
```

登录成功页面 Success.jsp 内容如下：

```
<%@ page language="java" import="java.util.*" pageEncoding="utf-8"%>
<%@ taglib uri="/struts-tags" prefix="s"%>
<%
    String path = request.getContextPath();
    String basePath = request.getScheme() + "://"
            + request.getServerName() + ":" + request.getServerPort()
            + path + "/";
%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
    <head>
        <base href="<%=basePath%>">
        <title>My JSP 'Success.jsp' starting page</title>
    </head>
    <body>
      <s:property value="username"/>
      <s:text name="success"/>
    </body>
</html>
```

登录失败页面 Error.jsp 内容如下：

```
<%@ page language="java" import="java.util.*" pageEncoding="utf-8"%>
<%@ taglib uri="/struts-tags" prefix="s"%>
<%
    String path = request.getContextPath();
    String basePath = request.getScheme() + "://"
            + request.getServerName() + ":" + request.getServerPort()
            + path + "/";
%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
    <head>
        <base href="<%=basePath%>">
        <title>My JSP 'Error.jsp' starting page</title>
    </head>
    <body>
      <s:text name="error"></s:text>
    </body>
</html>
```

上述页面中的 Struts2 标签都可以访问资源文件，表单标签<s:textfield name="" key=""/>中的属性key用于访问资源文件，<s:text name=""/>标签中的name属性用于访问资源文件。

实验八、Struts2 拦截器

实验目的：掌握 Struts2 的拦截器设计方法

实验类型：综合

实验要求：必做

实验内容：

1. 建立一个 Action 类 MyAction.java 和配置文件 Struts.xml；

- MyAction.java

```
package action;
import com.opensymphony.xwork2.ActionSupport;
public class MyAction extends ActionSupport {
    private int age;//年龄
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    @Override
    public String execute() throws Exception {
        // TODO Auto-generated method stub
        System.out.println("Action execute......");
        return SUCCESS;
    }
}
```

- struts.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
   "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
   "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
   <package name="action" extends="struts-default">
      <action name="test" class="action.MyAction">
         <result>Success.jsp</result>
      </action>
   </package>
</struts>
```

2. 建立一个拦截器类 HelloWorldInterceptor.java，其代码如下；

```
package interceptor;
import action.MyAction;
import com.opensymphony.xwork2.ActionInvocation;
import com.opensymphony.xwork2.interceptor.AbstractInterceptor;
```

```
public class HelloWorldInterceptor extends AbstractInterceptor {
   //拦截方法
    @Override
    public String intercept(ActionInvocation arg0) throws Exception {
        // 获得被调用的Action类
        Object action = arg0.getAction();
        //打印HelloWorld
        System.out.println("拦截器信息:HelloWorld!");
        //执行Action或调用下一个拦截器
        String result = arg0.invoke();
        //执行完action后提示
        System.out.println("Action执行完毕!");
        return result;
    }
}
```

3. 在 struts.xml 中加入拦截器的配置，见 struts.xml 内容；

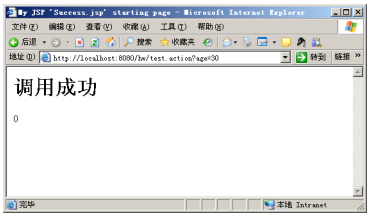
```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
   "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
   "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
   <package name="action" extends="struts-default">
      <!-- 定义拦截器 -->
    <interceptors>
      <interceptor name="helloworld"
class="interceptor.HelloWorldInterceptor"/>
    </interceptors>
      <action name="test" class="action.MyAction">
         <result>Success.jsp</result>
         <!-- action中引用拦截器 -->
         <interceptor-ref name="helloworld"/>
      </action>
   </package>
</struts>
```

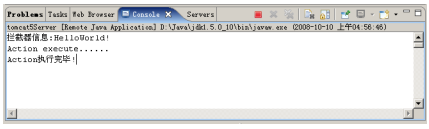
4. 编写 JSP 页面 Success.jsp。

```
<%@ page language="java" import="java.util.*" pageEncoding="utf-8"%>
<%@ taglib uri="/struts-tags" prefix="s" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>My JSP 'Success.jsp' starting page</title>
  </head>
  <body>
    <h1>调用成功</h1>
    <s:property value="age"/>
```

```
  </body>
</html>
```

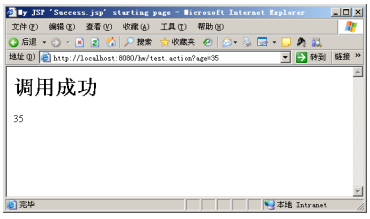
5. 运行该 web 程序，在地址栏输入 http://localhost:8080/hw/test.action?age=35。






```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
    "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
    <package name="action" extends="struts-default">
        <!-- 定义拦截器 -->
        <interceptors>
            <interceptor name="helloworld"
                class="interceptor.HelloWorldInterceptor" />
        </interceptors>
        <action name="test" class="action.MyAction">
            <result>Success.jsp</result>
            <!-- action中引用默认拦截器 -->
            <interceptor-ref name="defaultStack" />
           <!-- action中引用拦截器 -->
            <interceptor-ref name="helloworld" />
        </action>
    </package>
</struts>
```

6. 再次运行程序，输出结果正确。



- 文件上传

  上传单个文件的 JSP 页面代码如下：

```
<%@ page language="java" import="java.util.*" pageEncoding="utf-8"%>
<%@ taglib uri="/struts-tags" prefix="s" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
```

```
<html>
  <head>
    <title>My JSP 'index.jsp' starting page</title>
  </head>
  <body>
    <s:form action="fileupload" method="post" enctype="multipart/form-data">
    上传文件:<s:file name="doc"/><br>
    <s:submit value="上传"/>
    </s:form>
  </body>
</html>
```

form 表单的 enctype 属性设置为 multipart/form-data。enctype 用来指定表单数据的编码方式，有如下 3 个值。

1. application/x-www-form-urlencoded：指定该值，则表单中的数据被编码为 Key-Value 对，这是默认的编码方式。
2. multipart/form-data：使用 mime 编码，会以二进制流的方式来处理表单数据，文件上传需要使用该编码方式。
3. text/plain：表单数据以纯文本形式进行编码，其中不含任何控件和格式字符。

file 类型表单域 doc 用于选择上传文件，它和 Action 中的 java.io.File 类型的属性 doc 对应，同时上传文件的文件名对应于 Action 中的属性 docFileName，上传文件的文件类型对应于 Action 中的属性 docContentType。一般说来，为了上传文件，如果表单域名称为 xxx，那么在 Action 中应建立如下 3 个属性来接收上传文件的信息。

```
private java.io.File xxx;//封装上传文件的二进制内容
private String xxxContentType;//封装上传文件的文件类型
private String xxxFileName;//封装上传文件的文件名
```

封装上传文件的数据类 FileInfo.java 代码如下：

```
package po;
import java.io.File;
public class FileInfo {
    private File doc; //封装上传文件的属性
    private String docFileName; //封装上传文件的名称属性
    private String docContentType; //封装上传文件的类型属性
    private String targetdir; //保存路径
    private String targetfilename; //保存的文件名
    public File getDoc() {
        return doc;
    }
    public void setDoc(File doc) {
        this.doc = doc;
    }
    public String getDocContentType() {
        return docContentType;
    }
    public void setDocContentType(String docContentType) {
        this.docContentType = docContentType;
```

```
    }
    public String getDocFileName() {
        return docFileName;
    }
    public void setDocFileName(String docFileName) {
        this.docFileName = docFileName;
    }
    public String getTargetdir() {
        return targetdir;
    }
    public void setTargetdir(String targetdir) {
        this.targetdir = targetdir;
    }
    public String getTargetfilename() {
        return targetfilename;
    }
    public void setTargetfilename(String targetfilename) {
        this.targetfilename = targetfilename;
    }
}
```

用于文件上传的 Action 代码如下：

```
package action;
import java.io.File;
import org.apache.commons.io.FileUtils;
import org.apache.struts2.ServletActionContext;
import po.FileInfo;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.ModelDriven;

public class FileUploadAction extends ActionSupport implements
        ModelDriven<FileInfo> {
    // 封装文件信息
    FileInfo fileinfo = new FileInfo();
    public FileInfo getModel() {
        // TODO Auto-generated method stub
        return fileinfo;
    }
    @Override
    public String execute() throws Exception {
        // TODO Auto-generated method stub
        // 获得服务器上保存上传文件的目录updfile的绝对路径
        String realpath =
ServletActionContext.getServletContext().getRealPath(
                "/updfile");
        // 设置保存文件的目录
```

```
        fileinfo.setTargetdir(realpath);
        // 设置目标文件名

    fileinfo.setTargetfilename(generateFileName(fileinfo.getDocFileName()));
        // 把doc内容复制到target
        FileUtils.copyFile(fileinfo.getDoc(), new
File(fileinfo.getTargetdir(),
                fileinfo.getTargetfilename()));
        return SUCCESS;
    }
    // 产生唯一的文件名
    private synchronized String generateFileName(String filename) {
        int position = filename.lastIndexOf(".");
        String ext = filename.substring(position);
        return System.nanoTime() + ext;
    }
}
```

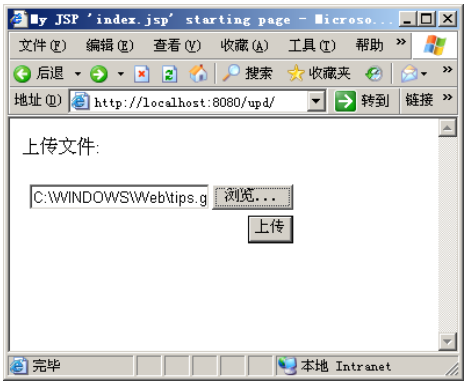
Action 的配置文件 struts.xml。

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
   "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
   "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
    <package name="action" extends="struts-default">
        <action name="fileupload" class="action.FileUploadAction">
           <result>Success.jsp</result>
        </action>
    </package>
</struts>
```

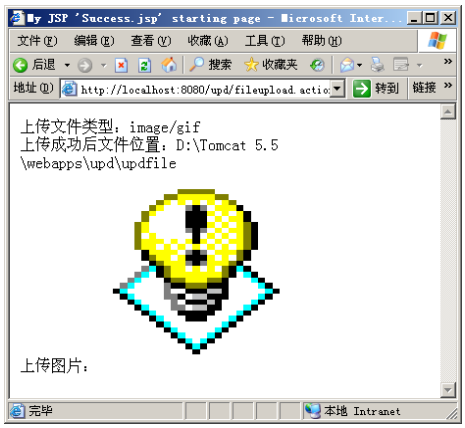
显示结果页面 Success.jsp。

```
<%@ page language="java" import="java.util.*" pageEncoding="GBK"%>
<%@ taglib uri="/struts-tags" prefix="s" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>My JSP 'Success.jsp' starting page</title>
  </head>
  <body>
    上传文件类型：<s:property value="docContentType"/><br>
    上传成功后文件位置：<s:property value="targetdir"/><br>
    上传图片：<img width=200 height=200 src="<s:property
value="'updfile/'+targetfilename"/>"><br>
  </body>
</html>
```

部署后运行，效果如图所示。



选择文件上传



图片上传后显示

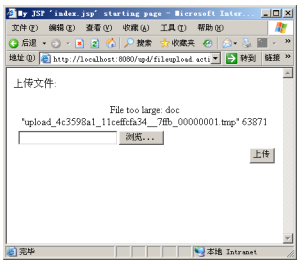
- 上传文件的过滤

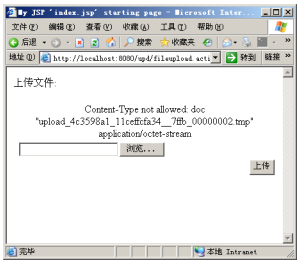
上传文件的时候可以限制上传文件的类型和大小，使用拦截器来实现。文件上传的拦截器是系统拦截器，我们只要配置参数就可以了。

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
    "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
    <!-- 上传文件的临时保存目录  -->
     <constant name="struts.multipart.saveDir" value="/tmp" />
     <package name="action" extends="struts-default">
         <action name="fileupload" class="action.FileUploadAction">
             <result>/Success.jsp</result>
             <result name="input">/index.jsp</result>
             <interceptor-ref name="fileUpload">
                 <!-- 允许上传的文件类型  -->
                 <param name="allowedTypes">
                     image/bmp,image/png,image/gif,image/jpeg
                 </param>
                 <!--上传文件的最大容量 单位字节  -->
                 <param name="maximumSize">20000</param>
             </interceptor-ref>
```

```
            <interceptor-ref name="defaultStack" />
        </action>
    </package>
</struts>
```

重新运行该程序，选择一个比较大的文件，单击“上传”按钮，结果如图所示。





错误提示都是英文，如果要改成中文提示，需要配置全局资源文件 globalMessage_zh_CN.proeprties，内容如下：

```
struts.messages.error.file.too.large=上传文件太大
struts.messages.error.content.type.not.allowed=上传文件必须是图像文件
struts.messages.error.uploading=上传过程出现异常
```

转换编码成为 unicode 编码。然后在 struts.xml 中引入该资源文件 <constant name="struts.custom.i18n.resources" value="globalMessage" />。

重新运行后，效果如图所示。

在 struts.xml 中配置<constant name="struts.multipart.saveDir" value="/tmp" />用于指定上传文件保存的临时目录，上传完成后系统会自动删除临时目录中的内容。

- 文件下载

文件下载可以通过配置 struts.xml 中 result 的类型来实现，执行下载操作的 Action 代码如下所示：

```
package action;
import java.io.InputStream;
import org.apache.struts2.ServletActionContext;
import com.opensymphony.xwork2.ActionSupport;
public class FileDownloadAction extends ActionSupport {
    private String inputpath; //下载文件路径
    private String contenttype; //文件类型
    private String filename; //文件名
    //返回一个InputStream类型
    public java.io.InputStream getInputStream() {
        return
ServletActionContext.getServletContext().getResourceAsStream(inputpath);
    }
    @Override
    public String execute() throws Exception {
        //调用相关业务逻辑方法 动态设置下载信息
        inputpath = "/updfile/Bliss.jpg";
        contenttype = "image/jpeg";
        //解决下载的中文文件名问题
        filename = java.net.URLEncoder.encode("文件.jpg","utf-8");
        return SUCCESS;
    }
    public String getContenttype() {
        return contenttype;
    }
    public void setContenttype(String contenttype) {
        this.contenttype = contenttype;
    }
    public String getFilename() {
        return filename;
    }
    public void setFilename(String filename) {
        this.filename = filename;
    }
    public String getInputpath() {
        return inputpath;
    }
    public void setInputpath(String inputpath) {
        this.inputpath = inputpath;
    }
}
```

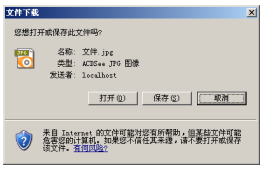
上述代码中的 java.io.InputStream getInputStream()方法必须有，它返回要下载文件的二进制输入流。

struts.xml 中配置下载 Action 的 result 类型为 stream，内容如下：

```
<action name="filedownload" class="action.FileDownloadAction">
            <result name="success" type="stream">
```

```
                <!-- 定义相关参数 -->
                <param name="contentType">${contenttype}</param>
                <param name="inputName">inputStream</param>
                <param name="bufferSize">4096</param>
                <param
name="contentDisposition">attachment;filename=${filename}</param>
            </result>
        </action>
```

地址栏输入 http://localhost:8080/upd/filedownload.action，出现如图所示画面。



下载图片