浙江工业大学 JavaEE技术实验报告

实验名称:Struts2 的控制器组件Action——登录用户的功能扩展

学院:计算机科学与技术学院、软件学院

班级： 软件工程1803

姓名： 石煌宽

组号： 无

时间： 2020/10/30

**一、实验主要步骤**

**一、基础实验——Struts2 框架搭建**

（一）实验目的

1、掌握Struts2 的Action 类中自定义方法的使用；

2、掌握Struts2 中Action 类的不同调用方式和相应的配置方法；

3、掌握Action 的实例化情况，理解Action 与Servlet 在实例化上的区别；

4、理解JSP 文件中获取Action 属性的过程；

5、了解Struts2 支持的Action 处理结束后的结果类型。

（二）基本知识与原理

1、Action 类中的默认方法名是execute()方法，可以被自动调用；

2、在Action 中也允许定义其它方法名，可以同时定义多个方法，分别处理不同的逻辑；

3、当Action 中使用了自定义方法，则该Action 就需要特定的配置，一般有四种调用方式：

（1）在struts.xml 文件中通过method 属性指定方法名；

（2）使用动态方法调用方式（DMI）；

（3）使用提交按钮的method 属性；

（4）使用通配符配置Action；

4、Action 类是多实例的，Action 类的属性是线程安全的；

5、在JSP 页面中，可以通过Struts2 标签调用Action 中对应的getter 方法，从而输出Action 的属性值；

6、当一个Action 处理用户请求结束后，返回一个字符串作为逻辑视图名，再通过struts.xml 文件中的配置将逻辑视图名与物理视图资源关联起来；Struts2

默认提供了一系列的结果类型（struts-default.xml 配置文件的result-types 标签里列出了所支持的结果类型），结果类型决定了Action 处理结束后，将调

用哪种视图资源来呈现处理结果。

（三）实验内容及步骤

1、在IDEA中新建Web 工程struts-prj2，并将Struts2 中的8 个核心包添加到工程中；

2、在struts-prj2 中新建login.jsp 页面，作为用户登录的视图；新建loginFail.jsp页面，作为登录失败的视图；新建loginSuccess.jsp 页面，作为登录成功的视图（可重用“实验二Struts 基础应用”中基础实验里的页面代码）；

3、在struts-prj2 中新建register.jsp 页面，作为用户注册的视图；新建regFail.jsp

页面，作为注册失败的视图；新建regSuccess.jsp 页面，作为注册成功的视图（可重用“实验二Struts 基础应用”中提高实验里的页面代码）；

4、在struts-prj2 中新建cn.edu.zjut.bean 包，并在其中创建UserBean.java，用于记录用户信息（ 可重用“ 实验二Struts 基础应用” 中提高实验里的UserBean.java 代码）；

5、在struts-prj2 中新建cn.edu.zjut.service 包，并在其中创建UserService.java，用于实现登录逻辑和注册逻辑（可重用“实验二Struts 基础应用”中提高实验里的UserService.java 代码）；

6、在struts-prj2 中新建cn.edu.zjut.action 包，并在其中创建UserAction.java，定义login()方法和register()方法，参照“实验二Struts 基础应用”写入代码，分别用于调用登录逻辑和注册逻辑（代码片段如下）；

package cn.edu.zjut.action;

……

public class UserAction {

……

public String login() {

UserService userServ = new UserService();

if (userServ.login(loginUser)) {

return "success"; }

return "fail";

}

public String register() {

UserService userServ = new UserService();

if (userServ.register(loginUser)) {

return "registersuccess"; }

return "registerfail";

}

}

7、在工程struts-prj2 的src 目录中创建struts.xml 文件，用于配置Action 并设置页面导航，通过action 标签中method 属性指定方法名（代码片段如下）；

<struts>

<package name="strutsBean" extends="struts-default" namespace="/">

<action name="login" class="cn.edu.zjut.action.UserAction"

method="login">

<result name="success">/loginSuccess.jsp</result>

<result name="fail">/loginFail.jsp</result>

</action>

<action name="register" class="cn.edu.zjut.action.UserAction"

method="register">

<result name="registersuccess">/regSuccess.jsp</result>

<result name="registerfail">/regFail.jsp</result>

</action>

</package>

</struts>

8、编辑Web 应用的web.xml 文件，增加Struts2 核心Filter 的配置；

9、将struts-prj2 部署在Tomcat 服务器上，通过浏览器访问login.jsp 与register.jsp页面，并记录运行结果；

10、查找相关资料，尝试使用Action 自定义方法的其它三种调用和配置方式：

动态方法调用方式（DMI）、提交按钮的method 属性、通配符配置Action，

并记录关键配置和运行结果；

重新配置sturts.xml,如下

<action name="UserAction" class="cn.edu.zjut.action.UserAction">

<result name="success">/loginSuccess.jsp</result>

<result name="fail">/loginFail.jsp</result>

<result name="register\_success">/regSuccess.jsp</result>

<result name="register\_fail">/regFail.jsp</result>

</action>

1. 使用动态方法调用方式（DMI）；

配置方法：声明action的名字和类型，如

<action name="UserAction" class="...UserAction">

</action>

访问方式：以actionName!methodName.action形式访问，如

<s:form action="UserAction!login" method="post">

（2）使用提交按钮的method 属性；

配置方法：声明action的名字和类型，如

<action name="UserAction" class="...UserAction"></action>

访问方式：以actionName!methodName.action形式访问，如

<s:submit value="登录" method="login"/>

1. 使用通配符配置Action；

11、修改UserAction.java，增加UserAction 类的构造方法UserAction()，增加count属性，用于测试Action 的实例化情况（代码片段如下）；

（四）实验要求

**二、提高实验——ActionSupport 与输入校验**

（一）实验目的

1、了解Action 接口的作用，理解ActionSupport 类的作用；

2、掌握在Struts2 中使用校验器或手工编码的方式，对请求参数进行数据校验的方法，掌握在JSP 页面中显示错误信息和提示信息的方法；

3、掌握在Action 中使用国际化资源文件的方法；

4、掌握Struts2 内置类型转换器的作用和使用方法。

（二）基本知识与原理

1、为了让用户开发的Action 类更规范，Struts2 提供了一个Action 接口，该接口定义了Struts2 的Action 处理类应该实现的规范；

2、Struts2 还为Action 接口提供了一个实现类：ActionSupport，该类提供了若干默认方法，包括：默认的处理用户请求的方法（excute()方法）、数据校验的方法、添加校验错误信息的方法、获取国际化信息的方法等。

3、Struts2 框架提供了校验器和手工编码两种方式对请求参数进行数据校验，当Action 类继承了ActionSupport 类，就可以通过定义名为“<ActionClassName>-<ActionAliasName>-validation.xml”的校验规则文件的方法进行校验器校验，也可以通过重写ActionSupport 类的validate()方法或validateXxx()方法进行手动校验；

4、在JSP 页面中使用Struts2 标签生成的表单，能将域级别的错误信息将自动显示到表单元素处；

5、在JSP 页面中使用fielderror 标签，可以集中显示所有的域级错误信息；使用actionerror 标签，可以显示所有的Action 级别错误信息；使用actionmessage标签，可以显示Action 消息；

6、Struts2 框架中提供了部分内置的类型转换器，可以将请求参数的String 类型转换成基本数据类型及对应的包装器类型、日期类型、数组类型、集合类型等，当Action 类继承了ActionSupport 类，则内置的类型转换器将默认生效，可以直接使用；

7、如需修改默认的类型转换校验信息，则要在Action 类的包中声明名为“Action类名.properties”的局部属性文件；

8、Struts2 框架同时支持自定义类型转换器，将请求参数转换成任意一种类型。

（三）实验内容及步骤

1、在struts-prj2 中修改UserAction 类，使其继承ActionSupport 类，并在UserAction类中覆盖ActionSupport 类的validate()方法，用于对用户登录的请求参数account 和password 进行校验：若用户名或密码为空，则使用addFieldError（域级）添加错误信息（代码片段如下）；

import com.opensymphony.xwork2.ActionSupport; ……

public class UserAction extends ActionSupport {

……

public void validate() {

String account = loginUser.getAccount();

String pwd = loginUser.getPassword();

if (account == null || account.equals("")) {

this.addFieldError("loginUser.account", "请输入您的用户名！");

}i

f (pwd == null || pwd.equals("")) {

this.addFieldError("loginUser.password", "请输入您的密码！");

}

}

}

2、修改struts.xml 文件，在Action 的配置中增加validate()方法校验出错时的页面导航（<result name="input">），代码片段如下：

<struts>

<package name="strutsBean" extends="struts-default" namespace="/">

<action name="login" class="cn.edu.zjut.action.UserAction"

method="login">

<result name="success">/loginSuccess.jsp</result>

<result name="fail">/loginFail.jsp</result>

<result name="input">/login.jsp</result>

</action>

……

</package>

</struts>

3、重新将struts-prj2 部署在Tomcat 服务器上，通过浏览器访问login.jsp 页面，观察并记录运行结果；

4、修改login.jsp 页面，在表单前增加fielderror 标签：<s:fielderror/>，再通过浏览器访问login.jsp 页面，观察并记录运行结果；

5、修改UserAction.java，在调用登录逻辑的login()方法中，对登录情况进行校验：若登录成功，使用addActionMessage()方法添加“登录成功！”的Action 提示消息，若登录失败，使用addActionError()方法添加Action 级别的错误信息（代码片段如下）；

public class UserAction extends ActionSupport {

……

public String login() {

UserService userServ = new UserService();

if (userServ.login(loginUser)) {

this.addActionMessage("登录成功！");

return "success";

} else {

this.addActionError("用户名或密码错误，请重新输入！");

return "fail";

}

}

6、修改login.jsp 页面，增加actionerror 标签（<s:actionerror/>）Action 级别的错误信息；修改loginSuccess.jsp，使用actionmessage 标签（<s:actionmessage/>）显示Action 提示消息；

7、修改struts.xml 文件中用户登录的页面导航设置，将登录失败时转向的页面从loginFail.jsp 修改为login.jsp；

8、重新将struts-prj2 部署在Tomcat 服务器上，通过浏览器访问login.jsp 页面，观察并记录运行结果；

9、在工程struts-prj2 中创建“UserAction-login-validation.xml”校验规则文件，使其与UserAction 类位于同一目录下，配置校验信息，使用校验器对请求参数进行校验（代码如下）；

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE validators PUBLIC

"-//Apache Struts//XWork Validator 1.0.2//EN"

"http://struts.apache.org/dtds/xwork-validator-1.0.2.dtd">

<validators>

<field name="loginUser.account">

<field-validator type="requiredstring">

<param name="trim">true</param>

<message>用户名不能为空</message>

</field-validator>

</field>

<field name="loginUser.password">

<field-validator type="requiredstring">

<param name="trim">true</param>

<message>密码不能为空</message>

</field-validator>

</field>

</validators>

10、重新将struts-prj2 部署在Tomcat 服务器上，通过浏览器访问login.jsp 页面，观察并记录运行结果；

11、参考“实验二Struts 基础应用”中的扩展实验，将login.jsp、loginSuccess.jsp、loginFail.jsp 三个页面进行国际化处理，把需要进行国际化的内容以键值对的形式写入资源文件message\_zh\_CN.properties 和message\_en\_US.properties中；

12、在资源文件中添加校验信息的键值对， 并使用native2ASCII 工具， 将

message\_zh\_CN.properties 重新编码，将中文字符都转化为unicode 码（代码

片段如下）；

#message\_en\_US.properties

#field error message

login.account.null=Please input your account!

login.password.null=Please input your password!

#action error message

login.error= Account or password error, please input again!

#action message

login.success=Login successfully!

13、在工程 struts-prj2 的 src 目录中创建 struts.properties 文件，通过它加载资源文件（具体代码如下）；

truts.custom.i18n.resources=cn.edu.zjut.local.message

struts.i18n.encoding=GBK

14、修改 UserAction.java，使用 ActionSupport 类的 getText()方法，获取国际化资 源文件中的信息（代码片段如下）；

public class UserAction extends ActionSupport { ……

public String login() {

UserService userServ = new UserService();

if (userServ.login(loginUser)) {

this.addActionMessage(this.getText("login.success"));

return "success";

} else {

this.addActionError(this.getText("login.error"));

return "fail";

}

public void validate() {

String account = loginUser.getAccount();

String pwd = loginUser.getPassword();

if (account == null || account.equals("")) {

this.addFieldError("loginUser.account",

this.getText("login.account.null"));

}

if (pwd == null || pwd.equals("")) {

this.addFieldError("loginUser.password",

this.getText("login.password.null"));

}

}

}

15、修改 UserAction-login-validation.xml，获取国际化资源文件中的信息（代码片 段如下）；

<validators>

<field name="loginUser.account">

<field-validator type="requiredstring">

<param name="trim">true</param>

<message key="login.account.null"/>

</field-validator>

</field> ……

</validators>

16、重新将 struts-prj2 部署在 Tomcat 服务器上，通过浏览器访问 login.jsp 页面，观察并记录运行结果；

17、修改 UserBean.java，将用于保存注册用户生日的变量类型改为 Date 类型，使用 Struts2 内置的类型转换器对请求参数进行校验；

18、重新将 struts-prj2 部署在 Tomcat 服务器上，通过浏览器访问 register.jsp 页面，当用户输入的生日不合法时，观察并记录运行结果；

19、在 工 程 struts-prj2 的 cn.edu.zjut.action 包 中 创 建 局 部 属 性 文 件“UserAction.properties”，修改类型转换的校验信息，并使用 native2ASCII工具将 UserAction.properties 重新编码（代码片段如下）；

#其中 invalid.fieldvalue 不能随意修改，loginUser.birthday 是请求参数域名，

#应根据实际需要进行修改

invalid.fieldvalue.loginUser.birthday=生日必须是日期，并符合

“yyyy-mm-dd”格式

20、重新将 struts-prj2 部署在 Tomcat 服务器上，通过浏览器访问 register.jsp 页 面，当用户输入的生日不合法时，观察并记录运行结果；

21、参考实验步骤 9，在工程 struts-prj2 的 cn.edu.zjut.action 包中创建 UserAction -register-validation.xml 文件，增加校验信息的配置，使用校验器对用户注册 的请求参数进行校验，要求注册时两次密码输入相同、email 地址格式符合要 求等；

22、重新将 struts-prj2 部署在 Tomcat 服务器上，通过浏览器访问 register.jsp 页 面，观察并记录运行结果；

23、修改 UserAction 类，将 validate()的方法名改为 validateLogin()，并增加 validateRegister()方法，参考实验步骤 1，使用手工编码方式对请求参数进行 数据校验；

24、重新将 struts-prj2 部署在 Tomcat 服务器上，通过浏览器访问 register.jsp 页 面，观察并记录运行结果。

（四）实验要求

1、填写并上交实验报告，报告中应包括：

（1）运行结果截图；

（2）根据实验步骤 1-7 和步骤 21-22，查找相关资料，总结 Action 类中 validate()方法和 validateXxx()方法的作用、使用时的要点或注意事项，总结在 JSP 页面中显示错误信息和提示信息的方法，并记录实验中相应的关键代码；

（3）根据实验步骤 8-9、19-20，总结使用校验器校验的方法；在 Struts2 的核心包 xwork-core2.3.15.1.jar\com\opensymphony\xwork2\validator\validators路径下找到 default.xml 文件，查找相关资料，总结校验规则文件中主要元素的作用和配置方法，并结合相应案例将其记录下来；

（4）根据实验步骤 10-14，总结在 Action 中使用国际化资源文件的步骤及方法，并结合实验中相应的关键代码将其记录下来；

（5）根据实验步骤 15-18，总结 Struts2 中常用的内置类型转换器及其使用方法；

（6）碰到的问题及解决方案或思考；

（7）实验收获及总结。

2、上交程序源代码，代码中应有相关注释。

**三、扩展实验——Struts2 的国际化**

（一）实验目的

1、掌握在 Action 中访问 Servlet API 的四种方法，理解四种方法的区别；

2、进一步理解 Action 与 Servlet 的区别；

3、进一步熟悉 Struts2 标签的使用方法；

（二）基本知识与原理

1、Struts2 框架中的 Action 类没有与任何 Servlet API 耦合，因此 Action 类可以脱离 Servlet 容器环境进行单元测试；

2、ActionContext 是 com.opensymphony.xwork2 包中的一个类，该类表示一个Action 运行时的上下文；

3、当 Action 类需要通过请求、会话或上下文存取属性时，可以通过 ActionContext类完成，也可以通过实现 Struts 提供的接口：RequestAware、SessionAware和 ApplicationAware 完成，而不必调用 Servlet API 中的 HttpServletRequest、HttpSession 和 ServletContext 对象，从而保持 Action 与 Servlet API 的解耦；

4、在 Action 类中直接访问 Servlet API，可以通过实现 Struts2 提供的接口：ServletContextAware、ServletRequestAware、ServletResponseAware 完成，也可以通过 ServletActionContext 工具类实现，但 Action 将与 Servlet API 直接耦合。

（三）实验内容及步骤

1、在 struts-prj2 中修改 UserAction 类，通过 ActionContext 获取请求、会话和上下文对象相关联的 Map 对象来实现存取属性的功能（代码片段如下）；

public class UserAction extends ActionSupport { ……

private Map request,session,application;

public String login() {

//获取 ActionContext 对象

//通过 ActionContext 对象获取请求、会话和上下文对象相关联的 Map 对象

request=(Map) ctx.get("request");

session=(Map) ctx.getSession();

application=(Map) ctx.getApplication();

//访问 application 范围的属性值

Integer counter = (Integer)application.get("counter");

if(counter==null)

counter=1;

else

counter=counter+1;

//设置 application 范围的属性

application.put("counter", counter);

UserService userServ = new UserService();

if (userServ.login(loginUser)) {

//设置 session 范围的属性

session.put("user", loginUser.getAccount());

//设置 request 范围的属性

request.put("tip", "您已登录成功");

return "success";

} else {

return "fail";

}

}

}

2、修改 loginSuccess.jsp 页面，从请求、会话和上下文对象中获取属性值并显示 在页面中（代码片段如下）；

<body>

本站访问次数为：<s:property value="#application.counter"/><br>

<s:property value="#session.user"/>，

<s:property value="#request.tip"/>

</body>

3、重新将 struts-prj2 部署在 Tomcat 服务器上，通过浏览器访问 login.jsp 页面， 观察并记录运行结果；

4、修改 UserAction 类，通过实现 Struts 提供的接口：RequestAware、SessionAware 和 ApplicationAware，获取请求、会话和上下文对象相关联的 Map 对象来实 现存取属性的功能（代码片段如下）；

public class UserAction extends ActionSupport implements RequestAware,SessionAware,ApplicationAware{ …… private Map request,session,application; public void setRequest(Map request) {this.request=request;

}

public void setSession(Map<String, Object> session) {

this.session=session;

}

public void setApplication(Map<String, Object> application) {

this.application=application;

}

public String login() {

Integer counter = (Integer)application.get("counter");

if(counter==null)

counter=1;

else

counter=counter+1;

application.put("counter", counter);

UserService userServ = new UserService();

if (userServ.login(loginUser)) {

session.put("user", loginUser.getAccount());

request.put("tip", "您已登录成功");

return "success";

} else {

return "fail";

}

}

}

5、重新将 struts-prj2 部署在 Tomcat 服务器上，通过浏览器访问 login.jsp 页面，观察并记录运行结果；

6、修改 UserAction 类，查找相关资料，尝试通过接口：ServletContextAware、ServletRequestAware、ServletResponseAware 直接访问 Servlet API，实现以上实验步骤 1-3 的相同功能，重新运行并记录结果，代码如下；

private HttpServletRequest request;

@Override

public void setServletRequest(javax.servlet.http.HttpServletRequest httpServletRequest) {

this.request = httpServletRequest;

}

private HttpServletResponse response;

@Override

public void setServletResponse(javax.servlet.http.HttpServletResponse httpServletResponse) {

this.response = httpServletResponse;

}

private ServletContext application;

@Override

public void setServletContext(javax.servlet.ServletContext servletContext) {

this.application = servletContext;

}

public String login() {

HttpSession session = request.getSession();

Integer counter = (Integer)application.getAttribute("counter");

if(counter==null)

counter=1;

else

counter=counter+1;

application.setAttribute("counter", counter);

UserService userServ = new UserService();

if (userServ.login(loginUser)) {

session.setAttribute("user", loginUser.getAccount());

request.setAttribute("tip", "您已登录成功");

return "success";

} else {

return "fail";

}

}

7、修改 UserAction 类，查找相关资料，尝试通过 ServletActionContext 工具类直接访问 Servlet API，实现以上实验步骤 1-3 的相同功能，重新运行并记录结果,实验代码如下；

public String login() {

HttpServletRequest request = ServletActionContext.getRequest();

HttpSession session = request.getSession();

ServletContext application = ServletActionContext.getServletContext();

Integer counter = (Integer) application.getAttribute("counter");

if (counter == null)

counter = 1;

else

counter = counter + 1;

application.setAttribute("counter", counter);

UserService userServ = new UserService();

if (userServ.login(loginUser)) {

session.setAttribute("user", loginUser.getAccount());

request.setAttribute("tip", "您已登录成功");

return "success";

} else {

return "fail";

}

}

8、尝试利用 Servlet API 添加购物车功能，在工程 struts-prj2 的 cn.edu.zjut.bean包中创建 Item.java 用于记录商品信息（代码片段如下）；

public class Item {

private String itemID;

private String name;

private String description;

private double cost;

public Item(String itemID, String name,

String description, double cost) {

setItemID(itemID);

setTitle(name);

setDescription(description);

setCost(cost);

}

//省略 getters/setters 方法

}

9、在工程 struts-prj2 的 cn.edu.zjut.bean 包中创建 ItemOrder.java 用于记录购物车中的一条商品信息和购买数量（代码片段如下）；

public class ItemOrder {

private Item item;

private int numItems;

public ItemOrder(Item item) {

setItem(item);

setNumItems(1);

}

//省略 getters/setters 方法

}

10、在工程 struts-prj2 的 cn.edu.zjut.bean 包中创建 ShopppingCart.java 用于记录 用户的购物车信息，为简化操作，在购物车的构造函数中加入商品信息（代 码片段如下）；

public class ShoppingCart {

private List itemsOrdered;

public ShoppingCart() {

itemsOrdered = new ArrayList();

Item item = new Item("book001", "JAVAEE 技术实验指导教程",

"WEB 程序设计知识回顾、" + "轻量级 JAVAEE 应用框架、"

+ "企业级 EJB 组件编程技术、" + "JAVAEE 综合应用开发.", 19.95);

ItemOrder itemorder = new ItemOrder(item);

itemorder.setNumItems(2);

itemsOrdered.add(itemorder);

}

public List getItemsOrdered() {

return (itemsOrdered);

}

public synchronized void addItem(String itemID) {}

public synchronized void setNumOrdered(String itemID,

int numOrdered) {}

}

11、修改 UserAction 类的 login 方法，当登录成功时，创建 ShoppingCart 对象并存入会话中；

12、修改 loginSuccess.jsp 页面，从会话中获取购物车信息并显示在页面中（代码片段如下）；

<body>

<table border=1>

<s:iterator value="#session.shoppingcart.itemsOrdered" >

<tr>

<th>编号</th><th>名称</th><th>说明</th><th>单价</th><th>数量</th>

</tr>

<tr>

<td><s:property value="item.itemID"/></td>

<td><s:property value="item.name"/></td>

<td><s:property value="item.description"/></td>

<td><s:property value="item.cost"/></td>

<td><s:property value="numItems"/></td>

</tr>

</s:iterator>

</table>

</body>

13、重新将 struts-prj2 部署在 Tomcat 服务器上，通过浏览器访问 login.jsp 页面，观察并记录运行结果。

（四）实验要求

1、填写并上交实验报告，报告中应包括：

（1）运行结果截图；

（2）根据实验过程，观察 Action 中访问 Servlet API 的四种方法，总结四种方法

的区别，并记录下来；

（3）整理购物车功能相关的关键代码，记录相应的运行结果或报错信息；

（4）碰到的问题及解决方案或思考；

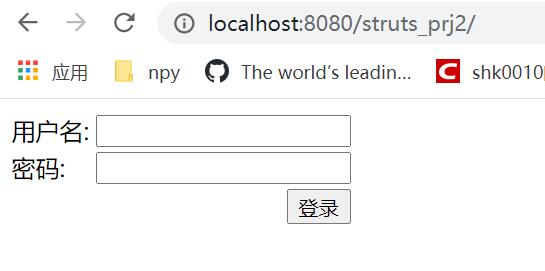
（5）实验收获及总结。

2、上交程序源代码，代码中应有相关注释。

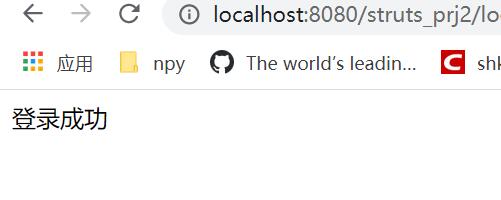
**二、实验结果及问题分析**

**实验一运行结果：**

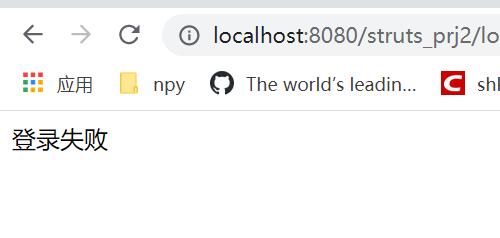
**登录系统**



**登录成功**



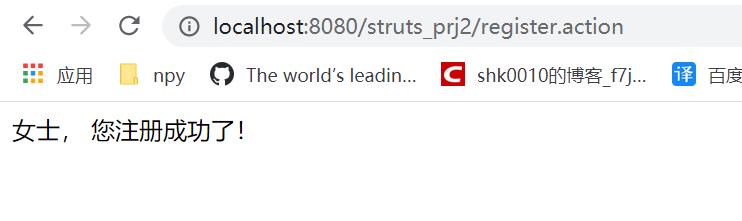
**登录失败**



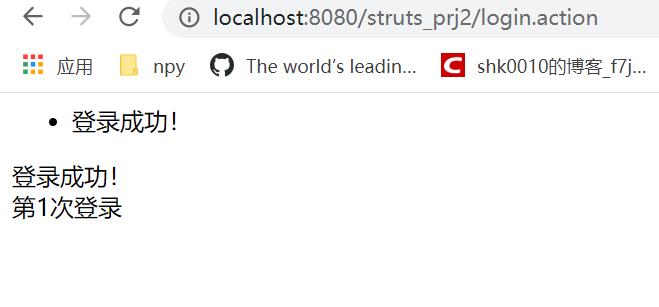
**注册系统**



**注册成功**



**登录成功2.0**



**登录成功3.0（重定向）**



**1）根据实验过程，总结 Action 自定义方法的四种调用和配置方式**

见实验一步骤10

**2）JSP 文件中获取 Action 属性的 主要过程**

使用标签直接调用即可。

**3）观察两次 loginSuccess.jsp 页面输出上的区别，分析原因并记录**

重定向并不能直接获取类内的值，但转发的可以。

**4）解压缩 Struts2 的核心包 struts2-core-2.3.15.1.jar，找到 struts-default.xml 配 置文件，在其中的 result-types 标签里列出了 Struts2 所支持的结果类型，查找相关资料，总结这些结果类型的作用和特点，并记录下来；**

如下所示：

<!-- 配置系统支持的结果类型 -->

<result-types>

<!-- Action链式处理的结果类型 -->

<result-type name="chain" class="com.opensymphony.xwork2.ActionChainResult"/>

<!-- 用于与JSP整合的结果类型-->

<result-type name="dispatcher" class="org.apache.struts2.dispatcher.ServletDispatcherResult" default="true"/>

<!-- 用于与FreeMarker整合的结果类型-->

<result-type name="freemarker" class="org.apache.struts2.views.freemarker.FreemarkerResult"/>

<!-- 用于控制特殊的HTTP行为的结果类型-->

<result-type name="httpheader" class="org.apache.struts2.dispatcher.HttpHeaderResult"/>

<!-- 用于直接跳转到其他URL的结果类型-->

<result-type name="redirect" class="org.apache.struts2.dispatcher.ServletRedirectResult"/>

<!-- 用于直接跳到其他action的结果类型-->

<result-type name="redirectAction" class="org.apache.struts2.dispatcher.ServletActionRedirectResult"/>

<!-- 用于向浏览器返回一个InputStream的结果类型-->

<result-type name="stream" class="org.apache.struts2.dispatcher.StreamResult"/>

<!-- 用于整合Velocity的结果类型-->

<result-type name="velocity" class="org.apache.struts2.dispatcher.VelocityResult"/>

<!-- 用于整合XML/XSLT的结果类型-->

<result-type name="xslt" class="org.apache.struts2.views.xslt.XSLTResult"/>

<!-- 用于显示某个页面原始代码的结果类型-->

<result-type name="plainText" class="org.apache.struts2.dispatcher.PlainTextResult" />

<!--将请求参数以form的形式提交到指定地点的结果类型-->

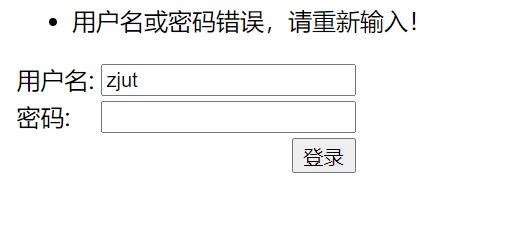
<result-type name="postback" class="org.apache.struts2.dispatcher.PostbackResult" />

</result-types>

**实验二实验结果：**

**登录成功、失败都成功实现与实验一结果相同**

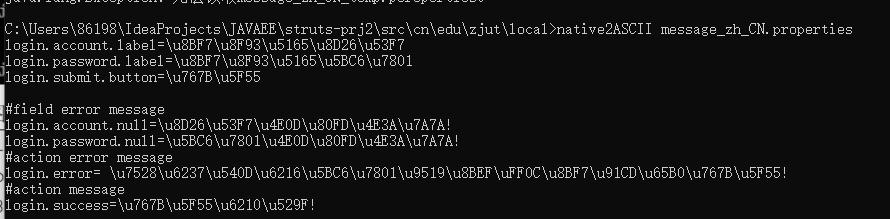
**登录失败2.0**



**错误信息1.0**



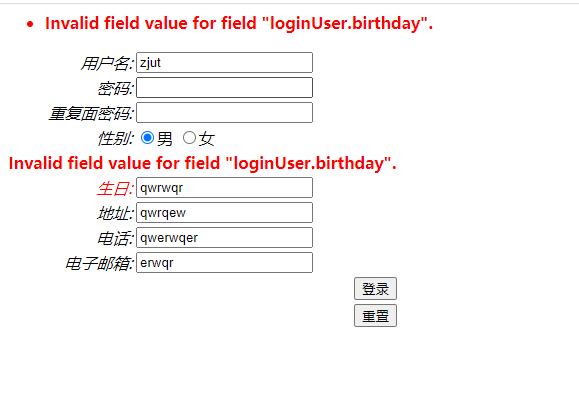
**国际化转码**



**国际化登录失败**



**生日格式校验报错**



**生日校验报错成功**



**注册报错1.0**



**注册校验2.0**



**1）总结 Action 类中 validate() 方法和 validateXxx()方法的作用、使用时的要点或注意事项，总结在 JSP 页 面中显示错误信息和提示信息的方法**

1.validate()方法会校验action中所有与execute方法签名相同的方法；

　　2.要校验指定的方法通过重写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；

　　3.当某个数据校验失败时，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；

　　4.在input视图中可以通过<s:fielderror/>显示失败信息。

5.先执行validateXxxx()->validate()->如果出错了，会转发<result name=”input”/>所指定的页面，如果不出错，会直接进行Action::execute()方法.

**2）总结使用校验器校验的方法**

<field name=校验对象>

<field-validator type=检验规则>

<!—校验规则-->

<message>校验信息</message>

</field-validator>

</field>

**3）在 Action 中使用国际化资源文件的步骤及方法**

校验器中：<message key="login.account.null"/>

validate方法中：this.addFieldError("loginUser.password", this.getText("login.password.null"));

**4）Struts2 中常用的内置类型转换器**

String

　　　　boolean / Boolean

char / Character int / Integer, float / Float, long / Long, double / Double

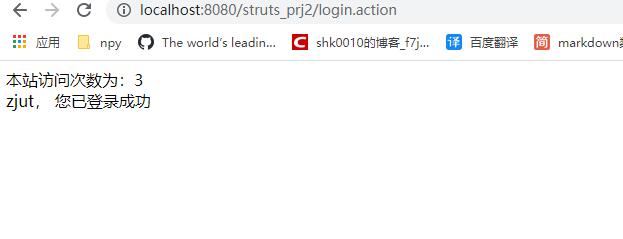
　　　　Date　　使用本地化的日期格式

　　　　arrays

　　　　collections

**实验三实验结果：**

**登录成功**



**后两组实验结果与第一次相同，皆成功调用参数**

**购物车显示**

1. **Action 中访问 Servlet API 的四种方法，总结四种方法 的区别**
   1. **通过ActionContext类完成**

//获取ActionContext对象

ActionContext ctx = ActionContext.getContext();

//通过ActionContext对象获取请求、会话和上下文对象相关联的Map对象

Map request=(Map) ctx.get("request");

Map session=(Map) ctx.getSession();

Map application=(Map) ctx.getApplication();

application.put("counter", counter);

* 1. **通过实现Struts提供的接口：RequestAware、SessionAware和ApplicationAware完成**

public class MyAction extends ActionSupport implements

RequestAware,SessionAware,ApplicationAware {

private Map request,session,application;

……

public void setRequest(Map<String, Object> request) {

this.request=request; }

public void setSession(Map<String, Object> session) {

this.session=session; }

……

}

* 1. **通过实现Struts提供的接口：RequestAware、SessionAware和ApplicationAware完成**

public class MyAction extends ActionSupport implements

RequestAware,SessionAware,ApplicationAware {

private Map request,session,application;

……

public void setRequest(Map<String, Object> request) {

this.request=request; }

public void setSession(Map<String, Object> session) {

this.session=session; }

……

}

* 1. **通过ServletActionContext工具类实现**

HttpServletRequest request =

ServletActionContext.getRequest();

HttpServletResponse response =

ServletActionContext.getResponse();

ServletContext application =

ServletActionContext.getServletContext();

HttpSession session = request.getSession();

……

application.setAttribute("counter", counter);

* 1. **通过ServletActionContext工具类实现**

HttpServletRequest request =

ServletActionContext.getRequest();

HttpServletResponse response =

ServletActionContext.getResponse();

ServletContext application =

ServletActionContext.getServletContext();

HttpSession session = request.getSession();

……

application.setAttribute("counter", counter);

**三、遇到的问题**

**问题1：**IDEA中途不能成功部署校验器文件

**解决方法**：以为是代码BUG，估计是IDEA什么参数炸了，直接酱将文件拖入classes文件夹中解决。

**问题2:** aplication参数不知道是什么

**解决方法：**查看ppt后了解是ServletContext的参数。

**四、实验总结及收获**

了解并运用了action自定义方法的四种调用和配置方式，检验器的使用及其国际化，struts的国际化， Action 中访问 Servlet API 的五种方法。对struts2的了解更加深刻了。