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： 计算机学院

班 级： 软件工程1801

姓 名： 陈南

学 号： 201806062101

组 号： 无

时 间： 2020年10月21日

# 一、实验主要步骤

**一、基础实验——Action 的自定义方法**

（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 Eclipse 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；

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）；

public class UserAction {

private Integer count=0;

**public UserAction(){**

**System.out.println("创建了一个 UserAction 类对象。");**

**}**

public Integer getCount() {

return count;

}

public String login() {

**count++;**

**//Action 实例化情况测试**

UserService userServ = new UserService();

if (userServ.login(loginUser)) {

return "success";

}

return "fail";

}

……

}

12、修改 loginSuccess.jsp，在页面中使用<s:property>标签输出 Action 中的 count

值；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，

并刷新多次，记录运行结果；

14、修改 struts.xml 文件，将 UserAction 的页面导航设置为 redirect 结果类型（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" **type="redirect"**>

/loginSuccess.jsp</result>

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

</action>

</package>

</struts>

15、重新将 struts-prj2 部署在 Tomcat 服务器上，通过浏览器访问 login.jsp 页面，

观察登录成功后 loginSuccess.jsp 页面的输出，并记录下来。

（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）查找相关资料，根据实验过程，总结 Action 自定义方法的四种调用和配置

方式；

（3）根据实验过程，记录实验步骤 11、12 中修改后的关键代码，及相应的运行

结果或报错信息，分析 Action 的实例化情况，将 Action 与 Servlet 在实例化

情况上进行对比，并记录下来；

（4）根据实验步骤 11、12，查找相关资料，分析 JSP 文件中获取 Action 属性的

主要过程，并记录下来；

（5）根据实验步骤 11-15，观察两次 loginSuccess.jsp 页面输出上的区别，分析原

因并记录下来；

（6）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；

（7）碰到的问题及解决方案或思考；

（8）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、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，部分重要方法

列表如下：

表 3-1 ActionSupport 类的部分重要方法

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>-

**方法名**

**说明**

public String execute()

默认的处理用户请求的方法，直接返回 SUCCESS

字符串。

public void validate()

空的输入校验方法，常被 Action 类覆盖，实现对

输入参数的校验。

public void addActionError(String

anErrorMessage)

将 Action 级别的错误信息添加到 Action 中。

public void addActionMessage(String

aMessage)

将 Action 级别的消息添加到 Action 中。

public void addFieldError(String

fieldName, String errorMessage)

将域级错误信息添加到特定的域中。

public String getText(String aTextName)

从国际化资源文件中获取属性值，其中参数是属

性文件的 key 值<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！");**

}

if (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）；

**struts.custom.i18n.resources=cn.edu.zjut.local.message**

struts.i18n.encoding=GBK14、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-core-2.3.15.1.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——Action 类与 Servlet API**

（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**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

**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<String, Object> 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；

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；

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。

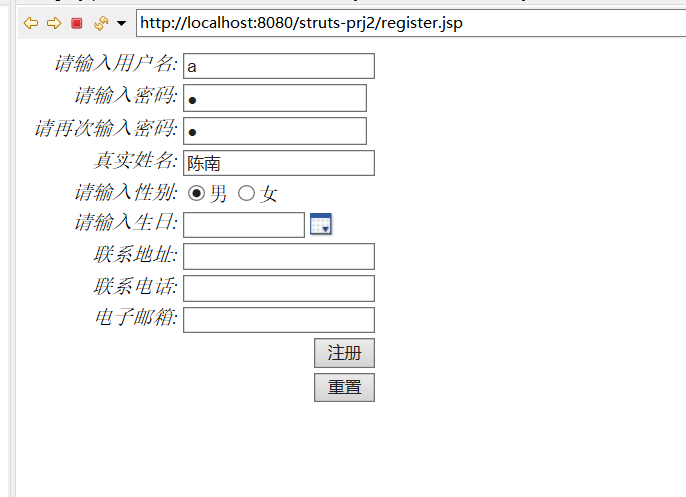
# 二、实验结果及问题分析

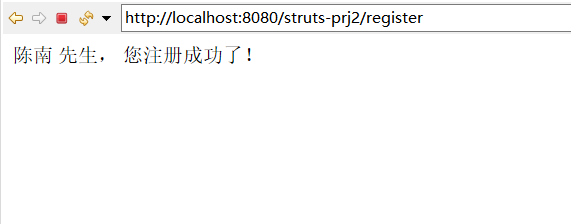
**一、基础实验——Action 的自定义方法**

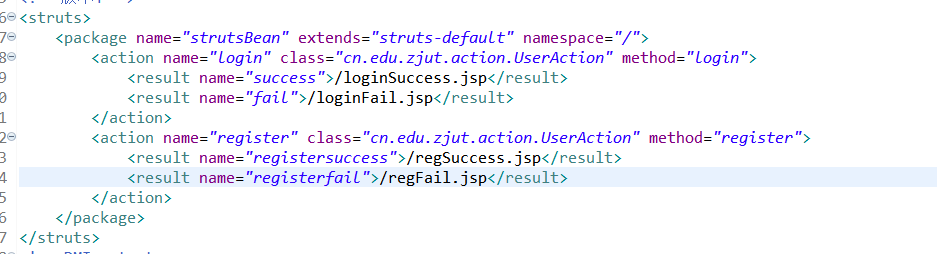
**（1）实验结果**

**1.普通的method配置方法**



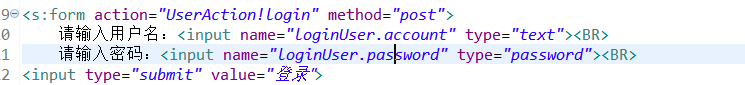


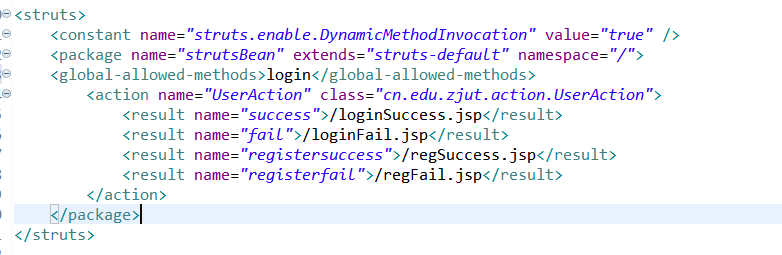


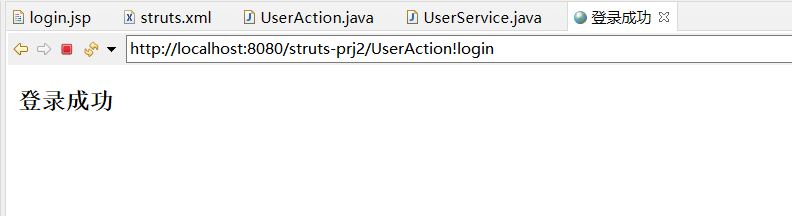




**2.动态方法调用方式（DMI） 配置及运行截图：**

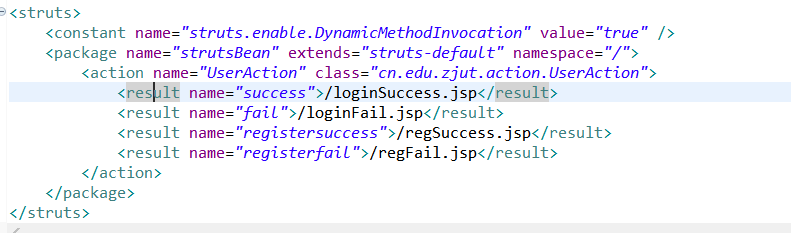


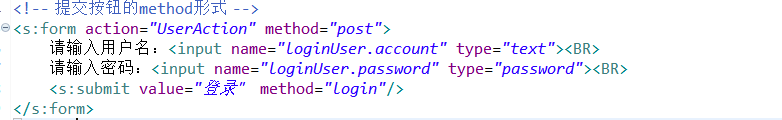


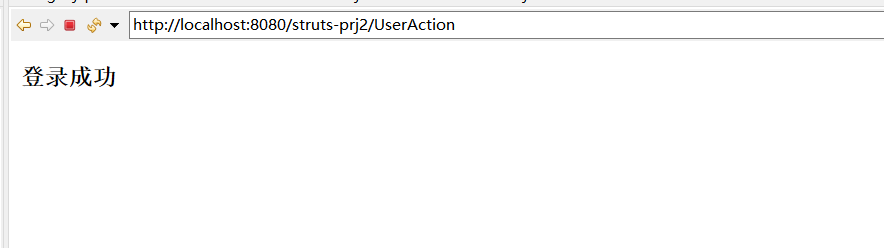


成功

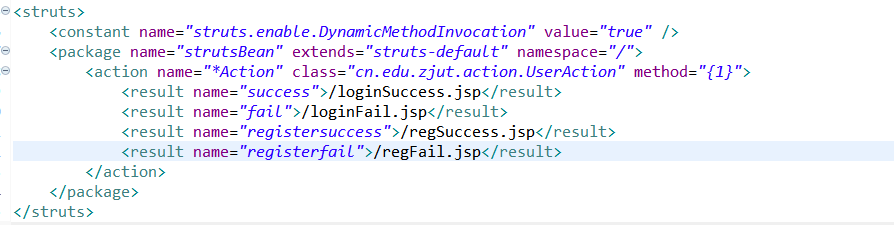
**3.提交按钮的 method 属性 配置及运行截图：**

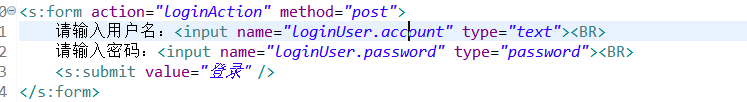






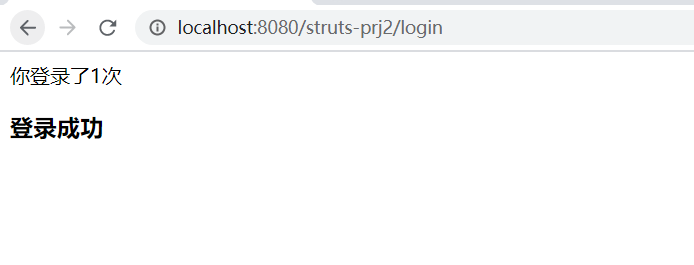
**4.通配符配置 Action2. 配置及运行截图：**



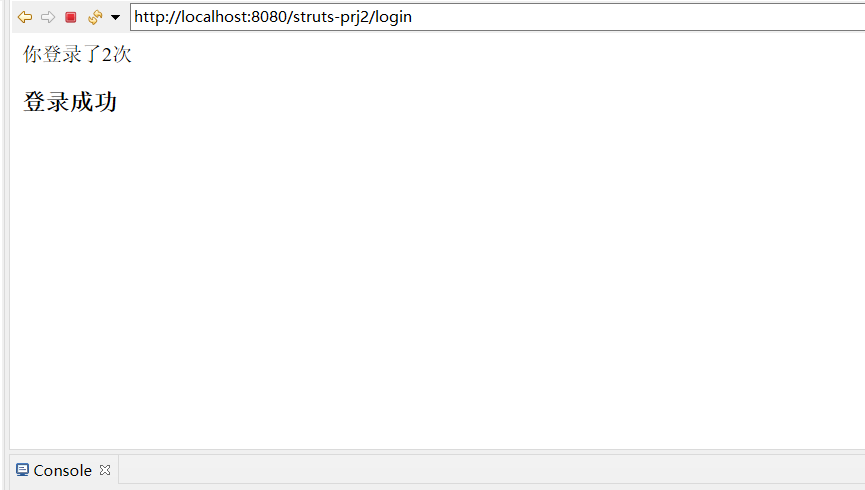




1. **增加 count 属性，用于测试 Action 的实例化情况**



**不管刷新多少次，他都会显示count的数值为1；**



**将count设置为static之后，在同一个网页上刷新会使得次数增多，在不同的网页上重新登录会重新变成一次。**

原因：与Action的生命周期有关，在第二次提交时，Action会重新实例化，这也是Action与Servlet之间的区别。

**6.将 UserAction 的页面导航设置为 redirect 结果类型**



发现不会显示count的值，原因在于用redirect作为重定向，原来的数据并不会一同带到之后的页面之中。

1. **Action 自定义方法的四种调用和配置方式**
2. 在struts.xml文件中通过method属性指定方法名

**关键代码：**

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

<result name="success" type="redirect">/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>

<form action="login" method="post">

请输入用户名：<input name="loginUser.account" type="text"><BR>

请输入密码：<input name="loginUser.password" type="password"><BR>

<input type="submit" value="登录">

</form>

1. 使用动态方法调用方式（DMI）

**关键代码：**

<struts>

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

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

<global-allowed-methods>login</global-allowed-methods>

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

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

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

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

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

</action>

</package>

</struts>

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

请输入用户名：<input name="loginUser.account" type="text"><BR>

请输入密码：<input name="loginUser.password" type="password"><BR>

<input type="submit" value="登录">

</s:form>

1. 使用提交按钮的method属性

**关键代码：**

<struts>

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

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

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

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

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

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

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

</action>

</package>

</struts>

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

请输入用户名：<input name="loginUser.account" type="text"><BR>

请输入密码：<input name="loginUser.password" type="password"><BR>

<s:submit value="登录" method="login"/>

</s:form>

1. 使用通配符配置Action

**关键代码：**

<struts>

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

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

<action name="\*Action" class="cn.edu.zjut.action.UserAction" method="{1}">

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

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

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

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

</action>

</package>

</struts>

<s:form action="loginAction" method="post">

请输入用户名：<input name="loginUser.account" type="text"><BR>

请输入密码：<input name="loginUser.password" type="password"><BR>

<s:submit value="登录" />

</s:form>

1. **根据实验过程，记录实验步骤 11、12 中修改后的关键代码，及相应的运行结果或报错信息，分析 Action 的实例化情况，将 Action 与 Servlet 在实例化 情况上进行对比，并记录下来**

**关键代码：**

public UserAction(){

System.out.println("创建了一个 UserAction 类对象。");

}

public Integer getCount() {

return count;

}

public String login() {

count++;//Action实例化情况测试

UserService userServ = new UserService();

if (userServ.login(loginUser)) {

return "success";

}

return "fail";

}

Action的生命周期与Servlet的生命周期是不同的；

例如用两个浏览器打开同一个网页，Action在第二次提交时，Action会重新进行实例化，而Servlet是在加载Servlet后，服务器创建一个Servlet实例（只执行一次）。

1. **根据实验步骤 11、12，查找相关资料，分析JSP文件中获取 Action 属性的主要过程，并记录下来；**
2. 可用EL表达式
3. 用Struts2自带的标签：

把要传的值定义为action的成员变量，为其建立set 、get方法，在action中为其赋值；然后在jsp中直接引用其名即可。

本题中运用的代码：

<s:property value="count"/>

1. **根据实验步骤 11-15，观察两次 loginSuccess.jsp 页面输出上的区别，分析原因并记录下来**
2. 不用redirect方式进行重定向

可以输出count的值

1. 用redirect的方式进行重定向

不能输出count的值

原因：不用redirect方式进行重定向，重定向的页面能够获取到action中的count值，只需要直接引用其名即可。用redirect的方式进行重定向的话，重定向中的页面无法获取到原action中的count

1. **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；**

1、chain：Action链式处理的结果类型

2、dispatcher：用于与jsp整合的结果类型，转发

3、freemarker：用于与freemarker整个的结果类型，

4、httpheader：用于控制特殊的Http行为的结果

5、redirect：用于重定向到其他URL的结果类型

6、redirectAction：用于重定向到其他Action的结果类型

7、stream：向浏览器返回一个InputStream的结果类型，在文件下载中会有用。

8、velocity：用于与velocity整个的结果类型。

9、xslt：用于与xml/xslt整个的结果类型。

10、plainText：返回页面原始代码

11、postback

**（7）碰到的问题，解决方案及思考**

1.在用DMI方式配置action时遇到的较大的困难：

（1）报如下错误



解决方法：

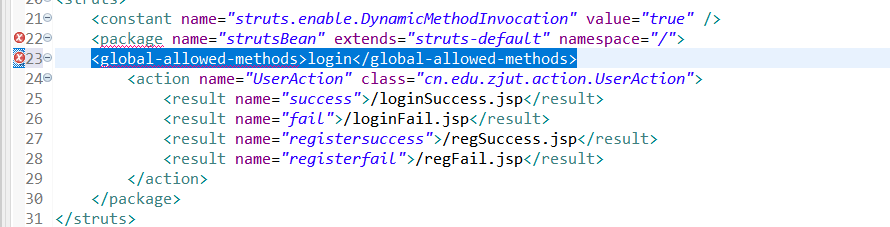
查阅相关资料之后，发现在配置struts.xml中要加入以下语句才能够运行



1. 在加入1的语句之后还是报错，报了一个不同的错误，如下（他说找不到login方法）：

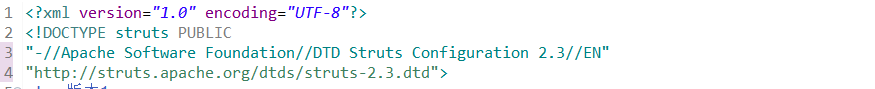


解决方法：再次去网上找了很多的资料，发现这是struts2 2.5版本中的问题，照着网上的解释加入了下列语句：



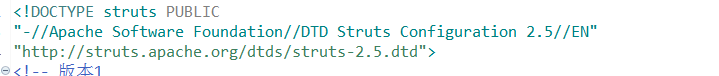
但是奇怪的是：他报错了！

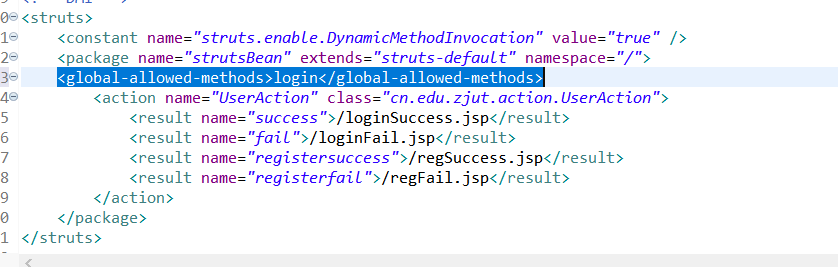
就很头疼，之后就重新检查了很多次代码，然后突然看到：



开头的几行由于是抄的pdf中的代码，没有自己修改过，还写的是2.3，然后在想有没有可能是这里的错误，把他改成了2.5……发现真的是这里的错误，大无语。

然后改成2.5之后他也不报错了，功能也能实现了，算是踩了一次坑，以后还是应该用老师用的包的版本……可以节约很多的时间，可以少踩很多不必要的坑。





(1)用提交按钮的method形式也报错……，报错内容：Wrong method was defined as an action method: execute

解决方法：网上没有找到2.5版本的相关修改方法，迫不得下载了2.3版本的包……错误就都解除了……

1. 还是要加入

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

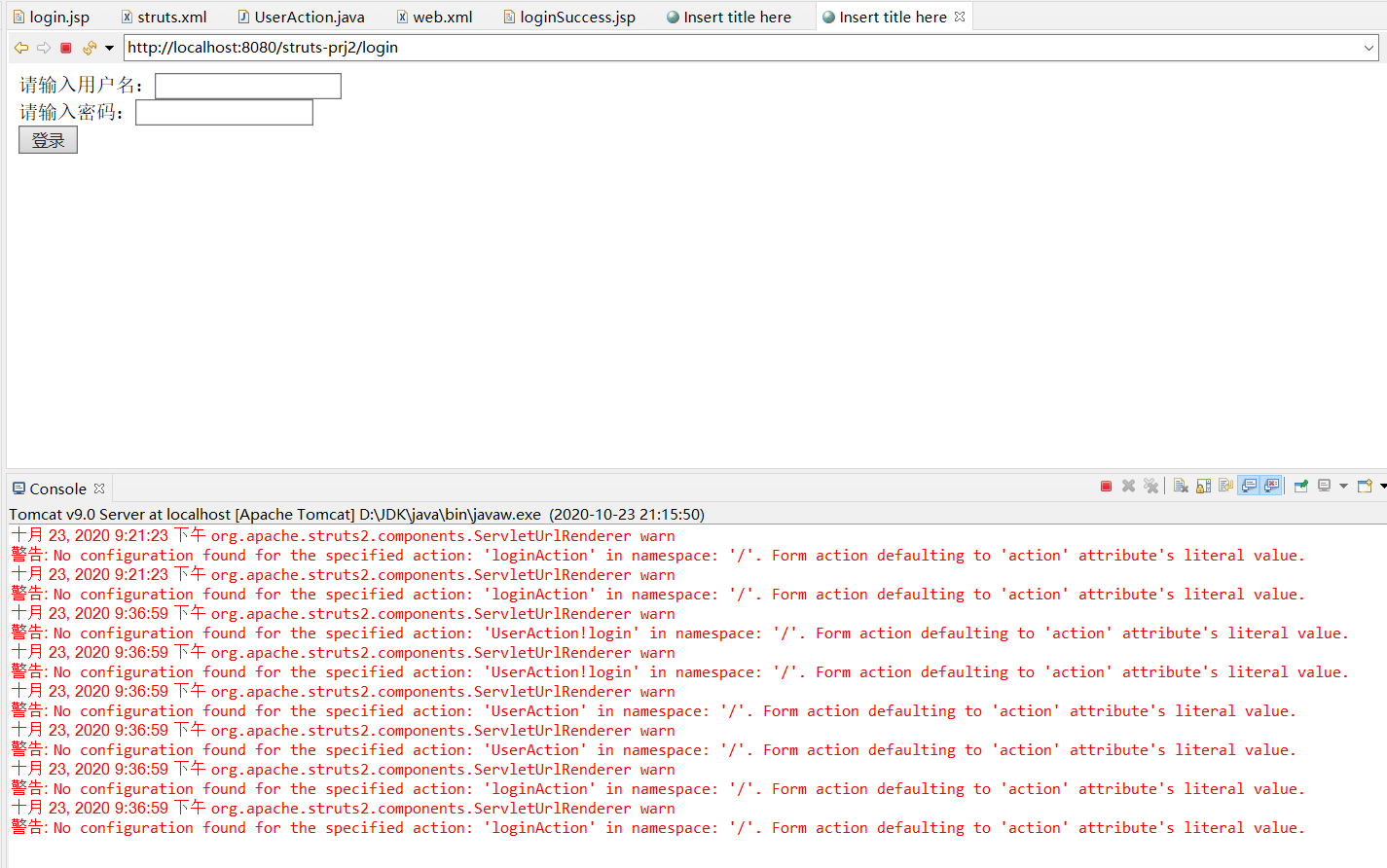
否则也无法正确运行

1. **实验收获及总结**
2. 熟悉并尝试了四种Action 自定义方法的配置方式，遇到了各种错误，有有关于版本的，有代码书写的问题，都在之后成功的解决了。
3. 了解明白了Action与Servlet生命周期之间的区别。
4. 了解将页面导航设置为 redirect后与普通的dispatch之间的区别。

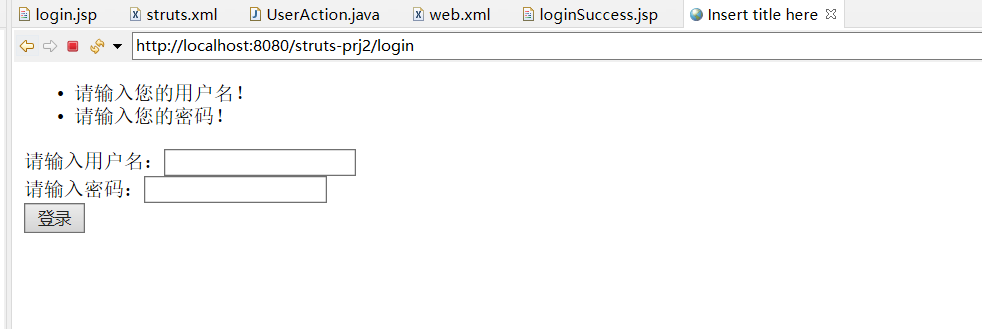
**二、提高实验——ActionSupport 与输入校验**

**（1）运行结果截图**

**1.未在表单前增加 fielderror 标签：<s:fielderror/>的login截图**



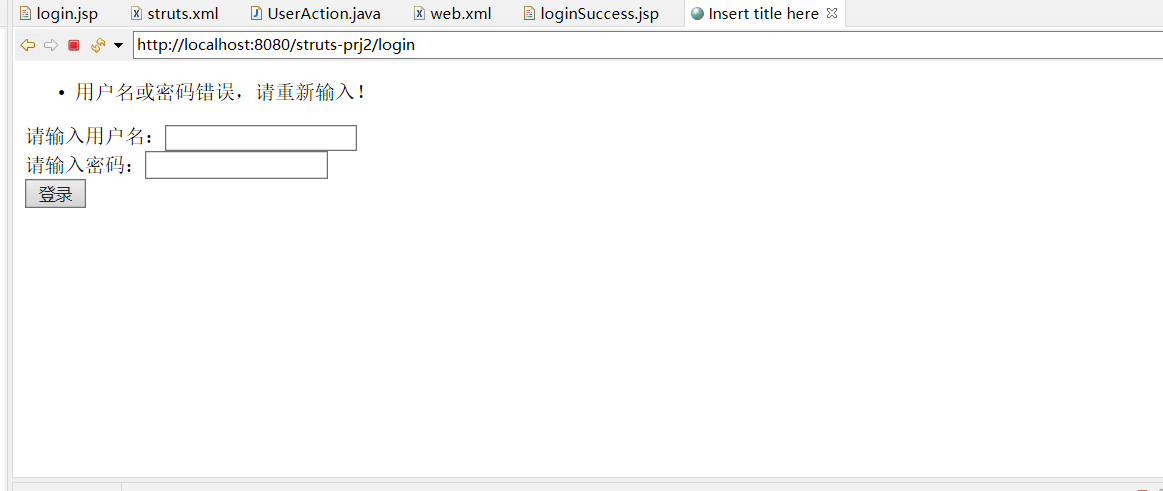
**2.在表单前增加 fielderror 标签：<s:fielderror/>的login截图后的实验结果**



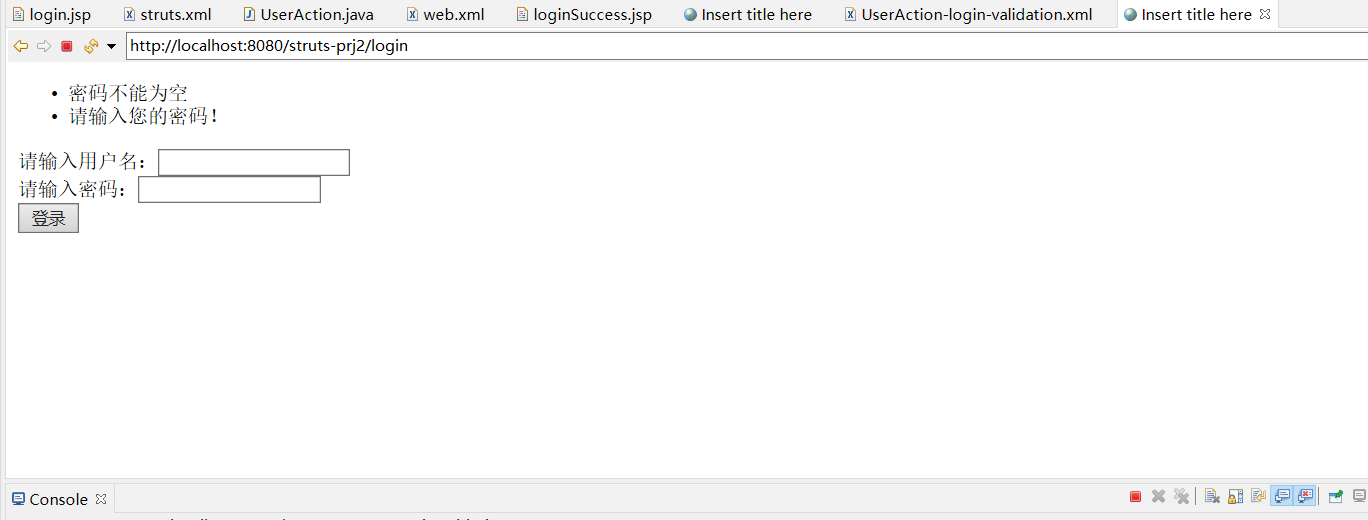
**3.对登录情况进行校验，login.jsp中用如下标签进行显示**

**<s:actionmessage/>**

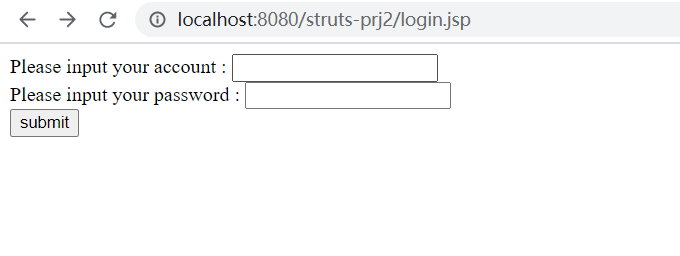
**<s:actionerror/>**



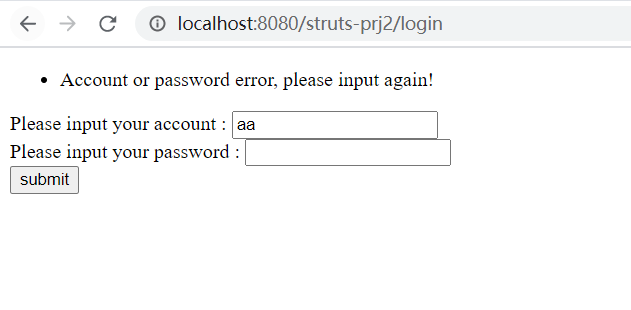
**4.创建完UserAction-login-validation.xml后**



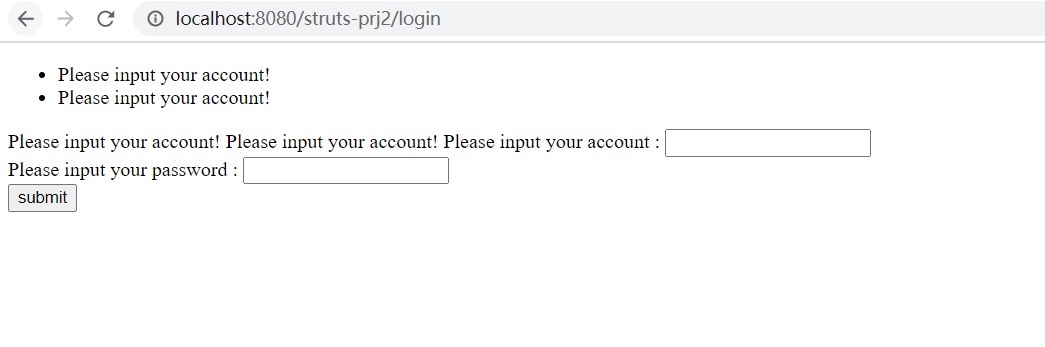
**5.国际化截图**



**用户名或密码错误：**



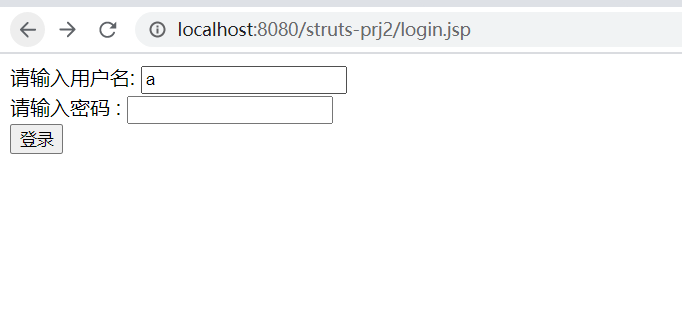
**用户名为空：**



**用户名或密码出错：**

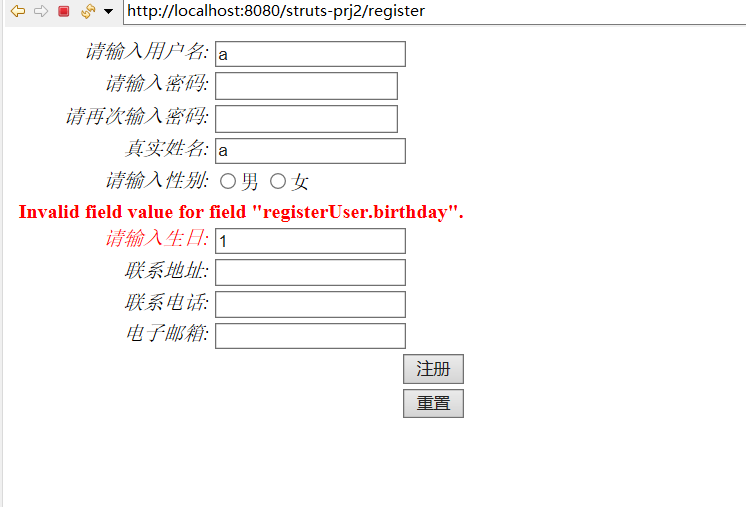
****

**密码为空：**

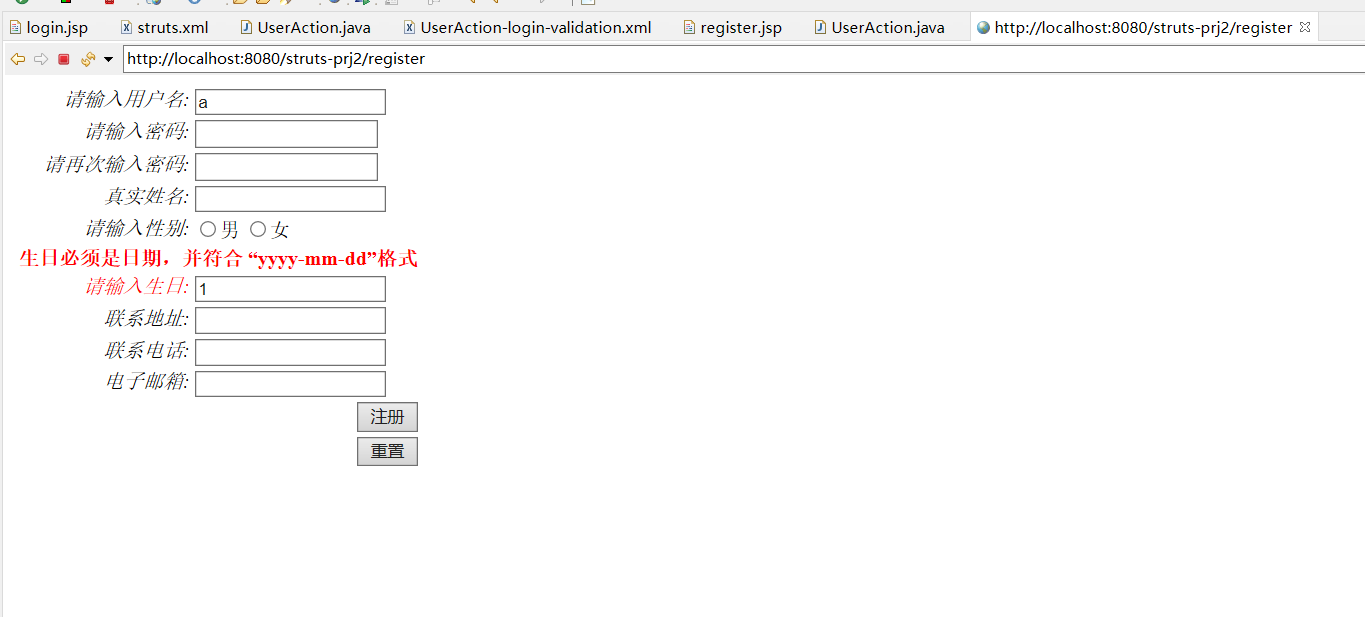




**6.未添加UserAction.properties**



**添加UserAction.properties**



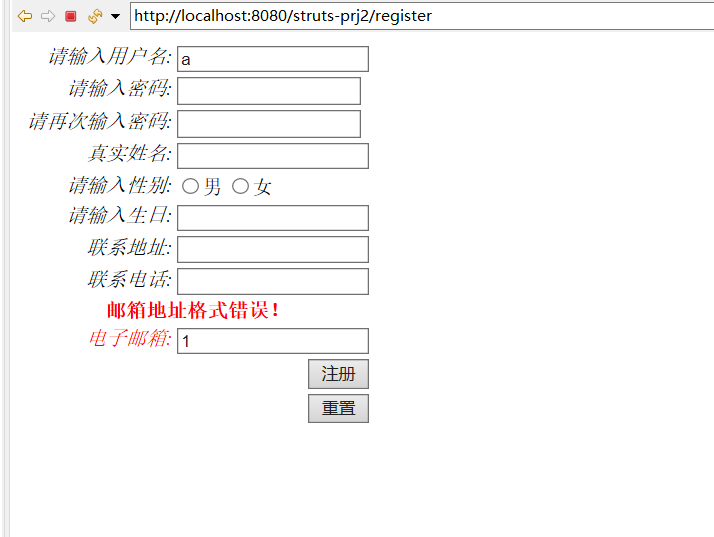
**输入正确格式可正常注册**

**6.使用校验器对用户注册的请求参数进行校验**

**两次密码输入不一致：**

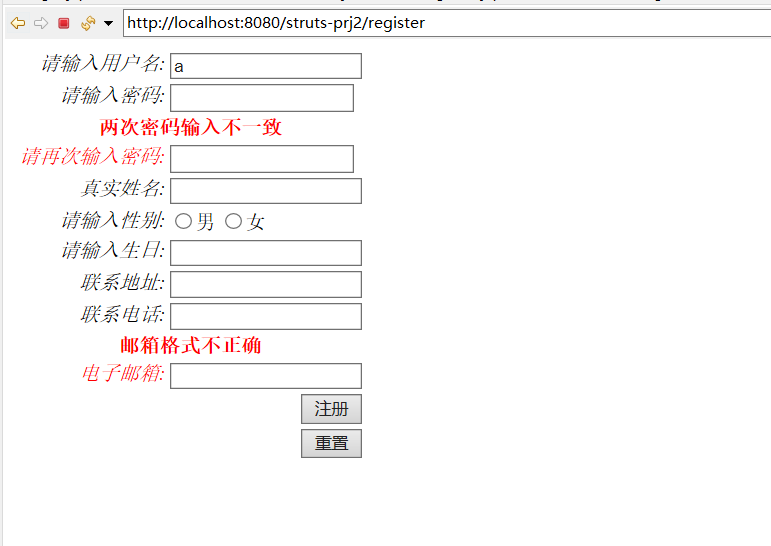


**邮箱格式不正确：**



1. **使手工编码方式对请求参数进行数据校验**

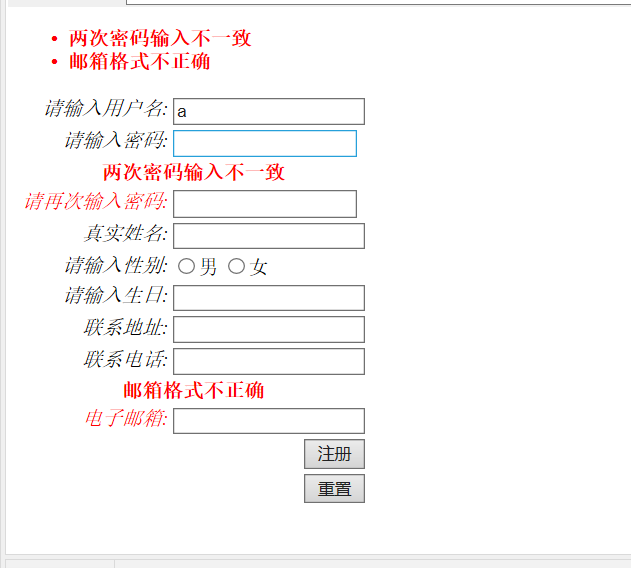
**register.jsp**



**login.jsp**



**在register.jsp中加入<s:fielderror/>**



**（2）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**

答：valiadate方法和validateXxx方法都是用来进行数据校验的，在发现不合法的 数据时，可通过过调用super.addFieldError将错误信息添加到验证框架中。注意点：validate的执行顺序是在execute方法之前的，是在set方法之后的，如果在执行validate方法时，没有通过数据校验，那么目标action方法将不再执行。如果在执行validate方法时，没有通过数据校验，那么将返回结果为input。所以一定要记得添加input的result节点。ValidateXxx方法是对action类中某一特定的action方法时添加，这种方法只验证特定的xxx方法。

注意：action类中同时有validateXxx（）和validate（）时，会先执行validateXxx（）方法。

在JSP中用一下代码显示错误和提示信息:

<s:actionmessage/>

<s:actionerror/>

<s:fielderror/>

关键代码：（validateXxx方法）

public void validateLogin() {

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"));

}

}

public void validateRegister() {

String account = registerUser.getAccount();

String pwd = registerUser.getPassword();

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

this.addFieldError("registerUser.account",this.getText("login.account.null"));

}

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

this.addFieldError("registerUser.password",this.getText("login.password.null"));

}

String repwd = registerUser.getRepassword();

String email = registerUser.getEmail();

if(!pwd.equals(repwd)) {

this.addFieldError("registerUser.repassword","两次密码输入不一致");

}

if(!dataPass(email,"^[a-zA-Z\_]{1,}[0-9]{0,}@(([a-zA-z0-9]-\*){1,}\\.){1,3}[a-zA-z\\-]{1,}$")) {

this.addFieldError("registerUser.email", "邮箱格式不正确");

}

}

private boolean dataPass(String str,String regEx)

{

Pattern pattern=Pattern.compile(regEx,Pattern.CASE\_INSENSITIVE);

Matcher matcher =pattern.matcher(str);

return matcher.matches();

}

1. **总结使用校验器校验的方法**

校验器校验的方法：配置XXXAction-Xxx-validate.xml文件进行校验器的书写。

实例：

<field name="loginUser.account">

<field-validator type="requiredstring">

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

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

</field-validator>

</field>

主要元素的作用和配置方法：

Required：必填校验器，要求字段必须有值

Requiredstring：必填字符串校验器，要求必须有值且长度大于0，即不能是空字符串。默认会去掉字符串前后空格

参数fieldName：该参数指定校验的字段名称，如果是字段校验，则不用指定该参数

参数trim：该参数为可选参数，用于指定是否在校验之前对字符串就行整理

Stringlength：字符串长度校验器，用于检验字段中字符串长度是否在指定的范围

可选参数mxnLength和minLength，分别用来指定最大或最小字符串长度。

Int：整数校验器，可以配置整数在指定的范围：

可选参数min和max，可分别指定字段值的最小值和最大值

Date：日期校验器，可以配置日期在指定的范围内

可选参数min和max，可分别指定字段日期值的最小值和最大值。

Email：邮件地址校验器，要求被检查的字段如果非空，贼必须是合法的邮件地址。

Regex：检查是否能匹配到正则表达式，参数为regex。

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,该逻辑表达式基于ValueStack进行求值,返回true时校验通过，否则不通过

Url：网址校验器,要求如果被校验的属性值非空,则必须是合法的url地址

配置方法：<field-validator>子元素

<message>错误信息</message>

<field-validator>子元素

<param name="">值</param>

用于指定校验器中的参数.

实例： <field name="registerUser.password">

<field-validator type="fieldexpression" short-circuit="true">

<param name="expression"><![CDATA[(registerUser.password == registerUser.repassword)]]></param>

<message>密码不一致，请重新输入。</message>

</field-validator>

</field>

用来判断注册页面两次输入的密码一致。

1. **总结在 Action 中使用国际化资源文件的步骤及方法**

国际化文件配置：同实验二中的扩展实验

关键代码：

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"));

}

在Action中运用getText函数，函数里面的内容去寻找配置文件中与之相匹配的字段，这样就可做到action的国际化。

1. **内置类型转换器**

内置类型转换器：默认生效，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。

常用的内置数据类型转换器：

String、boolean 、char / Character、 int / Integer、float / Float、 long / Long、 double / Double、Date　　使用本地化的日期格式、arrays 、collections

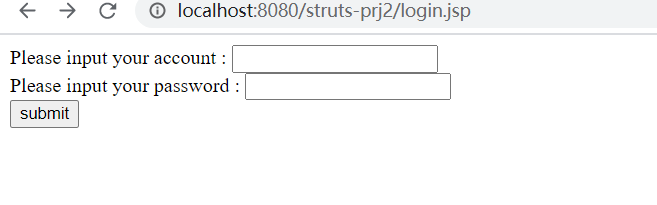
使用方法：

在UserBean中改变数据的类型，例如将生日改成date类型，这样虽然jsp中填入的是String类型的数据，但内置类型转换器能将生日数据自动转换成date类型。

**（6）问题及解决方案或思考  
1.在写国际化时遇到了button到了页面最右边的情况**



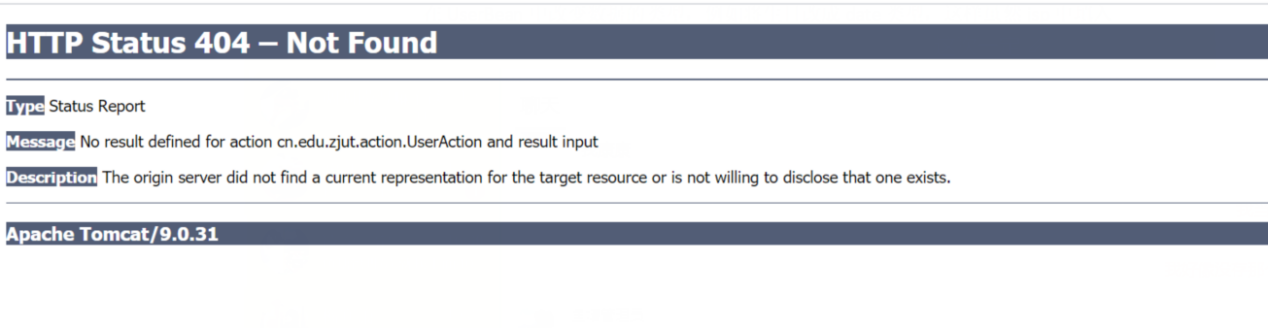
解决：将代码改成<s:submit theme = "simple" name="submit" key="login.submit.button"/>，添加了theme=”simple”这一串代码，成功将按钮移到了最左边



**2.一开始把内置类型转换器和校验器搞混了，没有分清楚两者分别能做什么工作，导致在写实验的时候遇到了较大的困难。**

解决方法：看ppt和去网上查阅相关资料，分清楚两者的区别、作用及用法，成功的完成了相应的配置之后实现了实验中的内容。

**3.没写input如下报错**



在struts.xml中添加如下语句

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

**（7）实验收获及总结**

1.在Action中配置了国际化，明白了相应的写法。

2.了解并实践了validate函数的用法、运行时间、作用等，和validateXxx函数进行了对比分析。

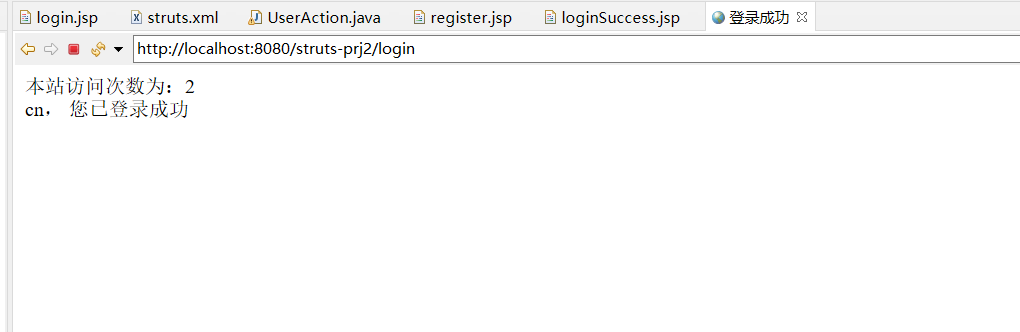
3.配置了校验器，实践了用校验器对jsp中的数据进行判断处理的操作。

4.尝试内置类型的转换器，成功的完成了string到date类型的转换，同时进行了相应的配置和匹配操作。

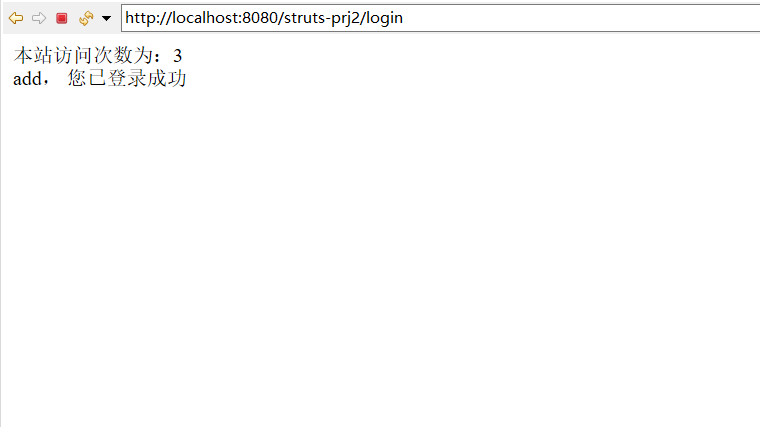
**三、扩展实验——Action 类与 Servlet API**

**（1）运行结果截图；**

**1.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**



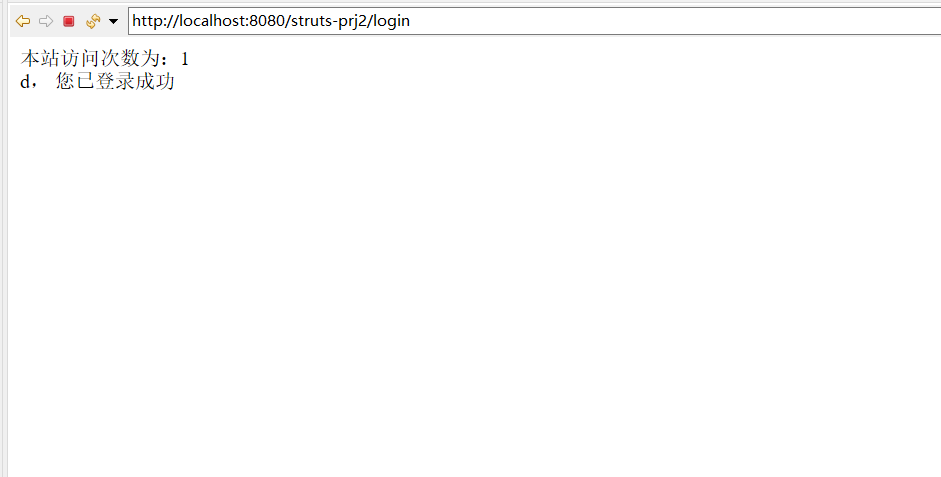
**2.通过实现 Struts 提供的接口：RequestAware、SessionAware ApplicationAware，获取请求、会话和上下文对象相关联的 Map 对象来实 现存取属性的功能**



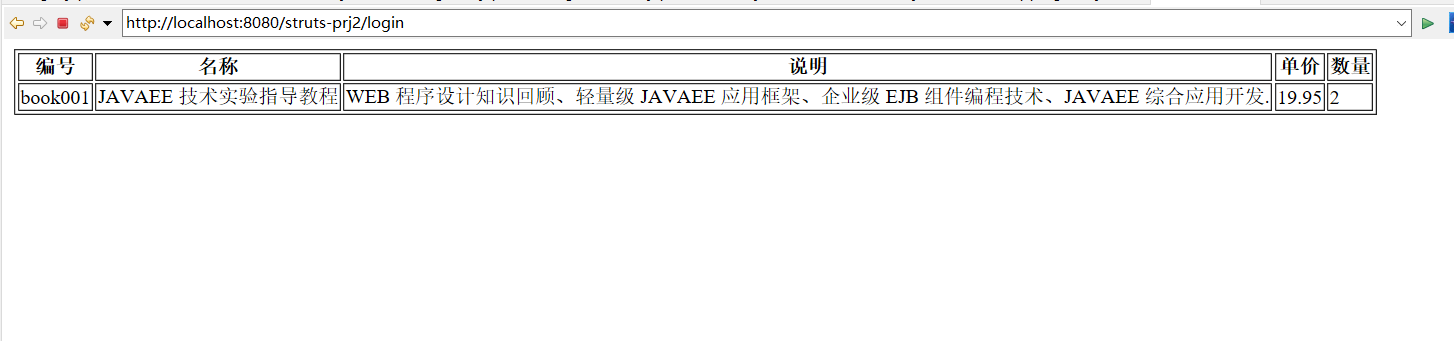
**3.尝试通过接口：ServletContextAware、 ServletRequestAwar、ServletResponseAware 直接访问 Servlet API**



**4.尝试通过ServletActionContent直接访问ServletAPI**



**5.购物车**



**（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；**

**1.ActionContext**

public static ActionContext getContext() ：获得当前Action的ActionContext实例。

public Object get(Object key) ：此方法类似于调用HttpServletRequest的getAttribute(String name)方法。

public void put(Object key, Object value) ：此方法类似于调用HttpServletRequest 的setAttribute(String name, Object o)。

public Map getParameters() ：获取所有的请求参数。类似于调用HttpServletRequest对象的getParameterMap() 方法。

public Map getSession() ：返回一个Map对象，该Map对象模拟了HttpSession实例。

public void setSession(Map session) ： 直接传入一个Map实例，将该Map实例里的key-value对转换成session的属性名－属性值对。

public Map getApplication() ：返回一个Map对象，该对象模拟了该应用的ServletContext实例。

public void setApplication(Map application) ：直接传入一个Map实例，将该Map实例里的key-value对转换成application的属性名－属性值对

**2.接口：RequestAware、SessionAware和ApplicationAware**

通过private Map<String,Object> request,session,application语句及set和get方法获取相关对象。

**3.接口：ServletContextAware、ServletRequestAware和ServletResponseAware**

通过以下语句和set和get方法获得对象。

private ServletContext application; // Servlet上下文

private HttpServletRequest request; // request对象

private HttpServletResponse response; // response对象

private Map session;

1. **ServletActionContext 工具类**

可直接通过以下语句获得对象

private HttpServletRequest request=ServletActionContext.getRequest();

private HttpServletResponse response=ServletActionContext.getResponse(); ServletContext application=ServletActionContext.getServletContext();

HttpSession session=request.getSession();

1. **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；**

**关键代码：**

private Map<String,Object> request,session,application;

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

ShoppingCart cart = new ShoppingCart();

UserService userServ = new UserService();

if (userServ.login(loginUser)) {

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

session.put("shoppingcart", cart);

return "success";

} else {

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

return "fail";

}

}

1. **碰到的问题及解决方案或思考；**
2. 在写实验时对Action的几种访问ServletAPI方法有点模糊和混淆，查阅网上的相关资料及实例代码之后才分清楚四种不同的写法。
3. 在购物车的书写上没遇到什么问题，可以去问问别人他们在书写时遇到了什么困难。

**（5）实验收获及总结**

1.熟悉明白并实践了Action的几种访问ServletAPI的方法，清楚了四种方法之间的区别与相似处。

2.实践了简单的购物车的实现。