浙江工业大学

JavaEE技术实验报告

实验名称：Struts2 的工作流程——登录用户的高级功能

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26日

**一、基础实验——拦截器与过滤器**

（一）实验目的

1、掌握 Struts2 自定义拦截器的基本开发步骤和配置方法；

2、掌握 Struts2 自定义过滤器的基本开发步骤和配置方法；

3、理解拦截器和过滤器的特点和区别；

4、了解 Struts2 默认拦截器栈中包含的主要拦截功能；

5、深入理解 Struts2 的工作原理和基本工作过程。

（二）基本知识与原理

1、Struts2 的控制器主要由三个层次组成，分别是过滤器、拦截器和业务控制器

Action；

2、过滤器是 Struts2 控制器的最前端控制器，过滤器的使用需要在 web.xml 中进

行配置；FilterDispatcher 是 Struts2 应用中必须配置使用的过滤器，该过滤器

的主要功能包括执行 Action、清空 ActionContext 对象等；

3、拦截器是 Struts2 中第二个层次的控制器，能够在 Action 执行前后完成一些通

用功能；

4、Struts2 内建了大量拦截器，这些拦截器以 name-class 对的形式配置在

struts-default.xml 文件中，如果 struts.xml 中定义的 package 继承了 Struts2 默

认的 struts-default 包，便可以直接使用默认拦截器栈 defaultStack；

5、Struts2 也允许自定义拦截器，自定义拦截器类须实现 Interceptor 接口，并覆

盖接口中的 intercept 方法用于实现拦截器的主要功能；自定义拦截器须在

struts.xml 中进行配置才能使用；

6、若在 struts.xml 中为 Action 指定了一个拦截器，则默认拦截器栈 defaultStack

将会失效，为了继续使用默认拦截器，必须将其进行显式地配置。

（三）实验内容及步骤

1、在 Eclipse 中新建 Web 工程 struts-prj3，并将 Struts2 中的 8 个核心包添加到工程中；

2、在 struts-prj3 中新建 login.jsp 页面，作为用户登录的视图；新建 loginSuccess.jsp

页面，作为登录成功的视图（可重用“实验三 Struts 的控制器组件 Action”

中基础实验里的页面代码）；

3、在 struts-prj3 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“实验三 Struts 的控制器组件 Action”中基础实验里

的 UserBean.java 代码）；

4、在 struts-prj3 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“实验三 Struts 的控制器组件 Action”

中基础实验里的 UserService.java 代码）；

5、在 struts-prj3 中新建 cn.edu.zjut.action 包，并在其中创建 UserAction.java，定

义login()方法用于调用登录逻辑（可重用“实验三 Struts的控制器组件Action”

中基础实验里的 UserAction.java 代码）；

6、在 struts-prj3 的 cn.edu.zjut.bean 中创建 Item.java，用于记录商品信息（可重

用“实验三 Struts 的控制器组件 Action”中扩展实验里的 Item.java 代码）；

7、在 struts-prj3 的 cn.edu.zjut.service 包中创建 ItemService.java，用于获取所有

商品信息，为简化操作，将商品信息直接写入代码中（代码片段如下）；

package cn.edu.zjut.service;

……

public class ItemService {

public List getAllItems(){

List items = new ArrayList();

items.add(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));

items.add(new Item("book002", "JAVAEE 技术",

"Struts 框架、Hibernate 框架、Spring 框架、"

+ "会话 Bean、实体 Bean、消息驱动 Bean", 29.95));

return items;

}

}

8、在 struts-prj3 的 cn.edu.zjut.action 包中创建 ItemAction.java，定义 execute()方

法用于调用“获取所有商品信息”逻辑（代码片段如下）；

package cn.edu.zjut.action;

……

public class ItemAction extends ActionSupport {

private List items;

//省略 getters/setters 方法public String getAllItems() {

ItemService itemServ = new ItemService();

items=itemServ.getAllItems();

System.out.println("ItemAction excuted!");

return "success";

}

}

9、在 struts-prj3 中创建 itemList.jsp 页面，作为显示所有商品信息的视图（代码

片段如下）；

<body>

<center>商品列表</center>

<table border=1>

<tr>

<th>编号</th><th>书名</th><th>说明</th><th>单价</th>

</tr>

**<s:iterator value="items" >**

<tr>

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

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

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

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

</tr>

**</s:iterator>**

</table>

</body>

10、修改 loginSuccess.jsp 页面，作为登录成功的视图，并在视图中增加超链接，

用于查看所有商品信息（代码片段如下）；

<a href="./allItems">查看所有商品信息</a>

11、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（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">/login.jsp</result>

</action>

<action name="allItems" class="cn.edu.zjut.action.ItemAction"

method="getAllItems">

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

</action>

</package></struts>

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

13、将 struts-prj3 部署在 Tomcat 服务器上，通过浏览器访问 login.jsp，登录成功

后点击超链接查看所有商品信息，观察并记录运行结果；

14、在 struts-prj3 中新 建 cn.edu.zjut.interceptors 包 ， 并在 其 中 创 建拦 截 器

AuthorityInterceptor.java，用于实现用户权限控制功能，使得只有登录用户才

有查看所有商品信息的权限（代码片段如下）；

package cn.edu.zjut.interceptors;

import java.util.Map;

import com.opensymphony.xwork2.Action;

import com.opensymphony.xwork2.ActionContext;

import com.opensymphony.xwork2.ActionInvocation;

import com.opensymphony.xwork2.interceptor.AbstractInterceptor;

public class AuthorityInterceptor **extends AbstractInterceptor**{

**public String intercept(ActionInvocation invocation)**

**throws Exception** {

System.out.println("Authority Interceptor executed!");

ActionContext ctx = invocation.getInvocationContext();

Map session = ctx.getSession();

String user = (String)session.get("user");

if(user!=null){

**return invocation.invoke();**

}

else{

ctx.put("tip", "您还没有登录，请输入用户名和密码登录系统");

/\*

ctx.getApplication().put("", "");//application 作用域

ctx.getSession().put("", "");//session 作用域

ctx.put("", ""); //request 作用域

\*/

**return Action.LOGIN;**

}

}

}

15、修改 UserAction.java，通过 ActionContext 获取 Session 对象相关联的 Map 对

象，当用户登录成功时，将用户名作为属性放入 session 范围内（代码片段如

下）；

public class UserAction extends ActionSupport {

……

private Map session;

public String login() {

**ActionContext ctx= ActionContext.getContext();session=(Map) ctx.getSession();**

UserService userServ = new UserService();

if (userServ.login(loginUser)) {

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

return "success";

} else {

return "fail";

}

}

}

16、修改 struts.xml 文件，增加拦截器的配置（代码片段如下）；

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

<！-- 定义一个名为 authority 的拦截器 -->

**<interceptors>**

**<interceptor name="authority"**

**class="cn.edu.zjut.interceptors.AuthorityInterceptor"/>**

**</interceptors>**

<action name="allItems" class="cn.edu.zjut.action.ItemAction"

method="getAllItems">

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

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

<！-- 配置系统默认拦截器 -->

**<interceptor-ref name="defaultStack"/>**

<！-- 配置 authority 拦截器 -->

**<interceptor-ref name="authority"/>**

</action>

……

</package>

17、重新将 struts-prj3 部署在 Tomcat 服务器上；首先不经用户登录直接通过浏

览器访问 loginSuccess.jsp 页面，点击超链接查看所有商品信息，观察并记录

运行结果；然后访问 login.jsp 页面，经用户登录后进入 loginSuccess.jsp 页面，

点击超链接查看所有商品信息，观察并记录运行结果；

18、在 struts-prj3 中 新 建 cn.edu.zjut.filters 包 ， 并 在 其 中 创 建 过 滤 器

AccessFilter.java，用于实现 JSP 页面的过滤功能，使得只有登录用户才能查看

除 login.jsp 和 register.jsp 之外的其它 JSP 页面（代码片段如下）；

package cn.edu.zjut.filters;

public class AccessFilter **implements Filter** {

……

**public void doFilter(ServletRequest arg0, ServletResponse arg1,**

**FilterChain filterChain) throws IOException, ServletException {**

System.out.println("Access Filter executed!");

HttpServletRequest request = (HttpServletRequest)arg0;HttpServletResponse response = (HttpServletResponse)arg1;

HttpSession session = request.getSession();

if(session.getAttribute("user")== null &&

request.getRequestURI().indexOf("login.jsp")==-1 &&

request.getRequestURI().indexOf("register.jsp")==-1){

**response.sendRedirect("login.jsp");**

return ;

}

**filterChain.doFilter(arg0, arg1);**

}

}

19、修改 web.xml 文件，增加过滤器的配置（代码片段如下）；

<web-app>

……

<filter>

<filter-name>accessFilter</filter-name>

<filter-class>cn.edu.zjut.filters.AccessFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>accessFilter</filter-name>

**<url-pattern>\*.jsp</url-pattern>**

</filter-mapping>

</web-app>

20、重新将 struts-prj3 部署在 Tomcat 服务器上；首先不经用户登录直接通过浏

览器访问 loginSuccess.jsp 和 itemList.jsp 页面，观察并记录运行结果；然后访

问 login.jsp 页面，经用户登录后进入 loginSuccess.jsp 页面，点击超链接查看

所有商品信息，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，查找相关资料，整理自定义拦截器类的作用和实现方法，

整理 Interceptor 接口中 intercept(ActionInvocation inv)、init()和 destroy()方

法的作用，并记录下来；记录实验步骤 14 中 intercept(ActionInvocation inv)

方法返回值的含义；

（3）根据实验过程，查找相关资料，整理自定义拦截器的配置步骤、注意事项，

并记录配置文件中相关标签的作用；（4）在 Struts2 核心包 struts2-core-2.3.15.1.jar 的 struts-default.xml 文件中找到

struts- default 包默认的拦截器栈 defaultStack 的定义，查找相关资料，整理

该拦截器栈中包含的主要拦截功能；

（5）根据实验过程，查找相关资料，整理自定义过滤器的实现方法和配置步骤，

将拦截器与过滤器进行比较，并将两者的特点及区别记录下来；

（6）根据实验过程，查找相关资料，总结 Struts2 的工作原理和基本工作过程，

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——值栈与 OGNL**

（一）实验目的

1、理解值栈的概念，了解值栈接口的主要方法和使用步骤；

2、掌握使用 OGNL 获取值栈内容的方法；

3、掌握使用 OGNL 获取 session、application 等其它对象的方法。

（二）基本知识与原理

1、Struts API 中的 com.opensymphony.xwork2.util.ValueStack 称为值栈，值栈是一

个数据区域，该区域中保存了应用范围内的所有数据和 Action 处理的用户请

求数据；

2、值栈被存储在 ActionContext 对象中，因此可以在任何节点访问其中的内容；

3、ValueStack 接口中主要方法有：Object findValue(String expr)可以通过表达式查

找值栈中对应的值，void setValue(String expr, Object value)用于将对象及其表

达式存到值栈中；

4、OGNL（Object Graphic Navigation Language），即对象图导航语言，是 Struts

默认的表达式语言；

5、OGNL 基础单位称为导航链，一个基本的链由属性名、方法调用、数组或集合

元素组成；

6、在 Struts2 中，值栈是 OGNL 上下文的根对象，可以直接访问，而 application、

session 等其它对象不是根对象，需要使用#进行访问。（三）实验内容及步骤

1、修改 itemList.jsp 页面，通过值栈对象获得属性；

（1）修改 itemList.jsp 页面的 page 指令，导入相关的 java 包，代码片段如下：

<%@ page language="java" contentType="text/html; charset=GB18030"

pageEncoding="GB18030"

**import="com.opensymphony.xwork2.util.ValueStack,**

**java.util.List,java.util.Iterator,**

**cn.edu.zjut.bean.Item"**%>

（2）修改<body></body>标签中的代码，获得值栈对象，通过值栈接口的

findValue 方法获得值栈中对象的值并输出，代码片段如下：

<body>

<%

**ValueStack vs=(ValueStack)request.**

**getAttribute("struts.valueStack");**

**String title=vs.findString("tableTitle");**

**List itemList=(List)vs.findValue("items");**

%>

<center>商品列表</center>

<table border=1>

<tr>

<th>编号</th>

<th>书名</th>

<th>说明</th>

<th>单价</th>

</tr>

<% Iterator it=itemList.iterator();

while(it.hasNext()){

Item item = (Item)it.next(); %>

<tr>

<td><%=item.getItemID() %></td>

<td><%=item.getTitle() %></td>

<td><%=item.getDescription() %></td>

<td><%=item.getCost() %></td>

</tr>

<%} %>

</table>

</body>

2、重新访问 login.jsp，登录成功后点击超链接查看所有商品信息，观察并记录

运行结果；

3、修改 itemList.jsp 页面，通过 OGNL 获得值栈内容，由于值栈是 OGNL 上下文的根对象，所以可以直接访问（代码片段如下）；

<body>

<center>商品列表</center>

<table border=1>

<tr>

<th>编号</th><th>书名</th><th>说明</th><th>单价</th>

</tr>

<s:iterator **value="items"** >

<tr>

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

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

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

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

</tr>

</s:iterator>

</table>

</body>

4、修改 itemList.jsp 页面，通过 OGNL 访问 session 对象，由于 session、application

等对象不是根对象，所以需要使用#进行访问（代码片段如下）；

<body>

<s:property **value="#session.user"**/>，您好！

<center>商品列表</center>

……

</body>

5、重新访问 login.jsp，登录成功后点击超链接查看所有商品信息，观察并记录

运行结果；

6、修改 itemList.jsp 页面，使用符号#过滤集合，取出价格小于 20 的商品和名称

为“JAVAEE 技术实验指导教程”的商品（代码片段如下）；

价格低于 20 元的商品有：<br>

<s:iterator value="**items.{?#this.cost<20}**" >

<li><s:property value="title"/>：

<s:property value="cost" />元</li>

</s:iterator>

<p>

名称为《JAVAEE 技术实验指导教程》的商品的价格为：

<s:property value="**items.**

**{?#this.title=='JAVAEE 技术实验指导教程'}.{cost}[0]**"/>元

7、修改 itemList.jsp 页面，使用符号%计算 OGNL 表达式的值，比较使用%和不使

用%的输出情况（代码片段如下）；

<s:url value="**items.{title}[0]**"/><br>

<s:url value="**%{items.{title}[0]}**"/>

8、重新访问 login.jsp，登录成功后点击超链接查看所有商品信息，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 1-2，查找相关资料，整理 ValueStack 接口及其主要方法的作

用和开发步骤，并记录下来；

（3）根据实验步骤 3-5，查找相关资料，整理 OGNL 可访问的对象和基本语法，

并记录下来；

（4）根据实验步骤 6-8，查找相关资料，整理 OGNL 三种常用符号：#、%和$的

作用和使用方法，并记录下来；

（5）碰到的问题及解决方案或思考；

（6）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 的异常处理**

（一）实验目的

1、掌握 Struts2 应用中处理异常的方式；

2、掌握在 struts.xml 中对 Action 类配置异常映射的方法。

（二）基本知识与原理

1、Struts2 应用中使用 Action 调用 Model，因此 Struts2 应用中的异常在 Model

层抛出后，通常在 Action 类中进行处理；

2、Action 可以直接使用 try/catch 捕获异常，然后返回结果视图，跳转到相关页

面处理异常；

3、抛出异常后，也可以不在 Action 类中捕获，而使用 throws 声明异常，交给

Struts2 框架处理；

4、Struts2 允许通过 struts.xml 文件来配置异常的处理，使用<exception-mapping>

标签声明异常映射，指定发生该类型异常时跳转的结果视图。

（三）实验内容及步骤1、在 struts-prj3 中新建 cn.edu.zjut.exception 包，并在其中创建自定义异常类

UserException.java，代码如下：

package cn.edu.zjut.exception;

public class UserException extends Exception{

public UserException() { super(); }

public UserException(String msg) { super(msg); }

public UserException(String msg, Throwable cause) {

super(msg, cause);

}

public UserException(Throwable cause) { super(cause); }

}

2、在 struts-prj3 中新建 loginException.jsp 页面，作为用户登录异常的视图，代

码片段如下：

<body> 登录异常！ </body>

3、修改 UserService.java，在 login 方法中，当用户名为 admin 时将抛出自定义异

常，当密码包含“ and ”或“ or ”时将抛出 SQLException，代码片段如下：

public class UserService {

public boolean login(UserBean loginUser) **throws Exception**{

**if (loginUser.getAccount().equalsIgnoreCase("admin")){**

**throw new UserException("用户名不能为 admin");**

**}**

**if (loginUser.getPassword().toUpperCase().contains(" AND ")**

**||loginUser.getPassword().toUpperCase().contains(" OR ")){**

**throw new java.sql.SQLException("密码不能包括' and '或' or '");**

**}**

if (loginUser.getAccount().equals(loginUser.getPassword())) {

return true;

}

else

return false;

}

……

}

4、修改 UserAction.java，在 login 方法中使用 try/catch 捕获异常，并在捕获异常

后返回结果视图，跳转到相关页面，代码片段如下：

public class UserAction extends ActionSupport {

……

public String login(){

……

UserService userServ = new UserService();

**try {**

if (userServ.login(loginUser)) {……

return "success";

} else {

……

return "fail";

}

**} catch (Exception e) {**

**e.printStackTrace();**

**return "exception";**

**}**

}

}

5、修改 struts.xml 文件，设置异常页面导航，代码片段如下：

<struts>

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

<interceptors>

<interceptor name="authority"

class="interceptors.AuthorityInterceptor"/>

</interceptors>

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

method="login">

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

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

**<result name="exception">/loginException.jsp</result>**

</action>

……

</package>

</struts>

6、将 struts-prj3 重新部署在 Tomcat 服务器上，通过浏览器访问 login.jsp，尝试

错误登录，观察并记录运行结果；

7、修改 UserAction.java，在 login 方法中抛出异常而不捕获，将异常交给 Struts2

框架处理，代码片段如下：

public class UserAction extends ActionSupport {

……

public String login() **throws Exception** {

……

UserService userServ = new UserService();

**try {**

if (userServ.login(loginUser)) {

……

return "success";

} else {……

return "fail";

}

**} catch (Exception e) {**

**throw e;**

**}**

}

}

8、修改 struts.xml 文件，使用<exception-mapping>标签完成异常配置，并通过全

局和局部两种方式进行异常映射，代码片段如下：

<struts>

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

<interceptors>

<interceptor name="authority"

class="interceptors.AuthorityInterceptor"/>

</interceptors>

**<global-results>**

**<result name="sqlExcp">/loginException.jsp</result>**

**</global-results>**

**<global-exception-mappings>**

**<exception-mapping exception="java.sql.SQLException"**

**result="sqlExcp"/>**

**</global-exception-mappings>**

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

method="login">

**<exception-mapping result="userExcp"**

**exception="cn.edu.zjut.exception.UserException"/>**

**<result name="userExcp">/loginException.jsp</result>**

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

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

</action>

……

</package>

</struts>

9、修改 loginException.jsp 页面，使用 Struts2 标签输出异常信息，代码片段如下：

<body>

异常信息：**<s:property value="exception.message"/>**

</body>

10、将 struts-prj3 重新部署在 Tomcat 服务器上，通过浏览器访问 login.jsp，尝试

错误登录，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 1-6，查找相关资料，整理自定义异常类的方法和步骤，并记

录下来；

（3）根据实验步骤 1-6，将 Action 使用 try/catch 捕获异常并返回结果视图的关

键代码和相关配置记录下来；

（4）根据实验步骤 7-10，查找相关资料，整理 Struts2 框架处理异常的机制，整

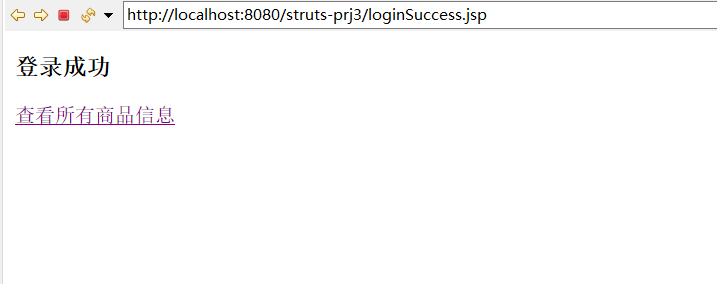
理 struts.xml 文件配置异常映射的方法以及相关标签的作用，并记录下来；

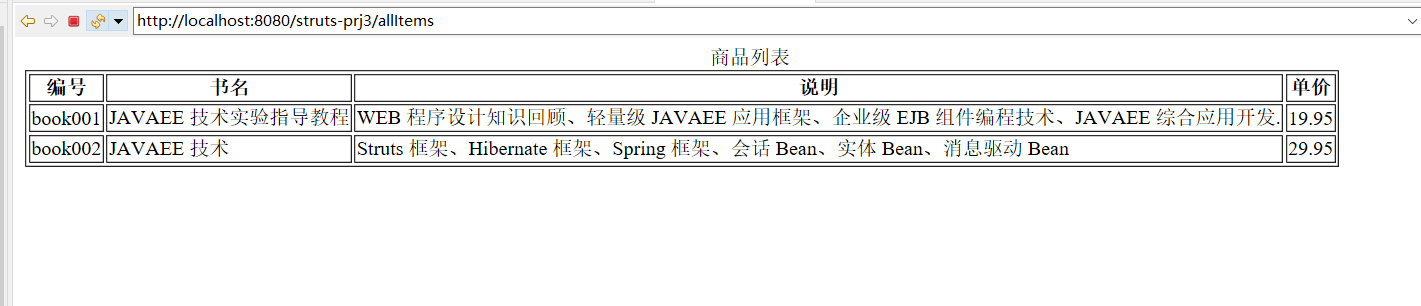
（5）碰到的问题及解决方案或思考；

（6）实验收获及总结。

2、上交程序源代码，代码中应有相关注释

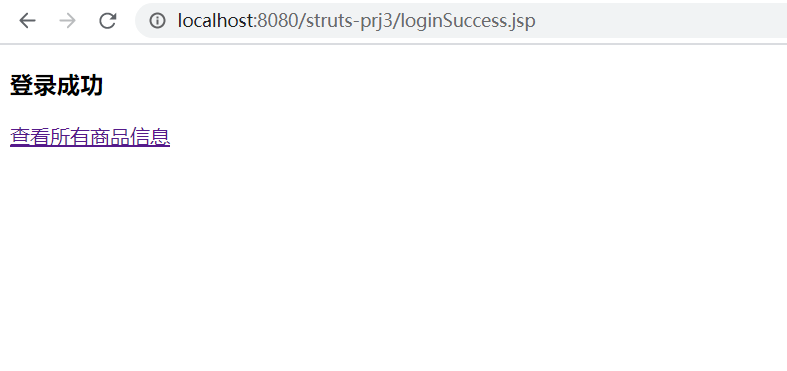
1. **基础实验——拦截器与过滤器**
2. **运行结果截图**
3. **点击链接查看所有的商品信息**



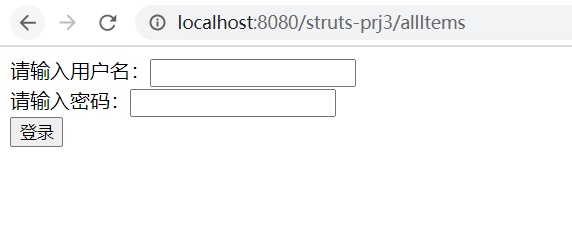


1. **基础实验17小题**

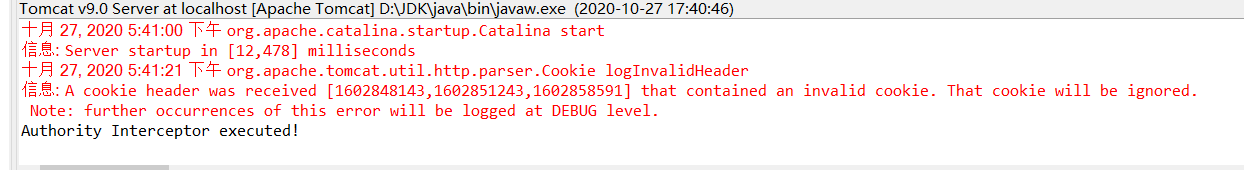
**【1】没有登录，直接进入到loginSucces.jsp页面**



**【2】单击超链接，返回到login.jsp中，url为allitems，**



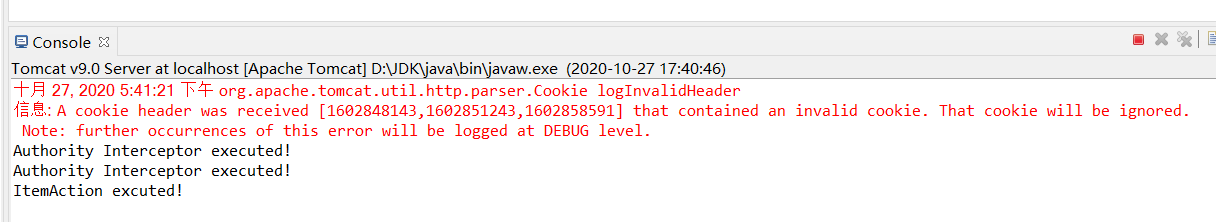
**工作台显示：**



**【3】访问login.jsp文件进行登录之后单击超链接，进入到allitems中并且输出书籍信息。**

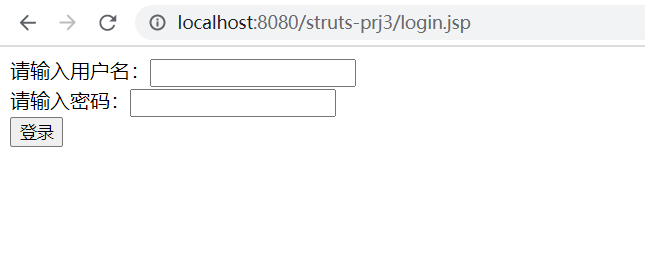


**工作台显示：**

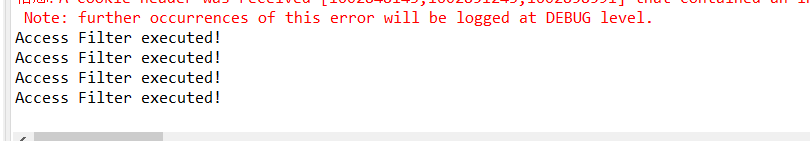


1. **基础实验20小题**

**【1】未登录的情况直接通过浏览器访问 loginSuccess.jsp 和 itemList.jsp 页面。**

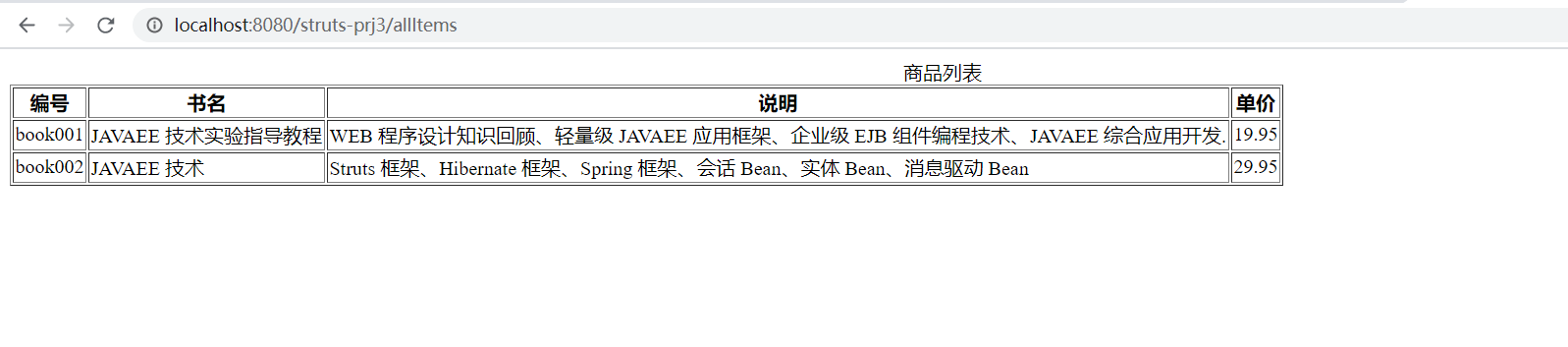


**工作台显示：**

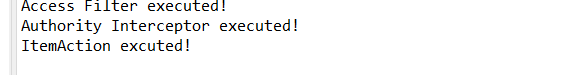


发现过滤器会阻止未登录的用户访问loginSuccess.jsp、itemList.jsp页面，并且会转到login.jsp页面。

**【2】登录之后能正常访问 loginSuccess.jsp 和 itemList.jsp 页面。**



**工作台显示：**



1. **自定义拦截器类的作用和实现方法。Interceptor接口中三种方法的作用。及intercept（ActionInvocation inv）方法返回值的含义。**

自定义拦截器类的作用和实现方法：

作用：拦截器可以让你在Action和bairesult被执行之前或之后进行一些处理。同时，拦截器也可以让你将通用的代码模块化并作为可重用的类。

所有的Strut2拦截器都会直接或间接的实现com.opensymphony.xwork2.

interceptor.Interceptor，并且需要实现其中的三种方法：分别为void init()

Void destroy()、String intercept(ActionInvocation invocation) throws Exception。在使用时，需要在struts.xml中注册这个拦截器。

核心代码：

<interceptors>

<interceptor

name="authority" class="cn.edu.zjut.interceptors.AuthorityInterceptor"/>

</interceptors>

<!-- 配置系统默认拦截器 -->

<interceptor-ref name="defaultStack"/>

<!-- 配置 authority 拦截器 -->

<interceptor-ref name="authority"/>

三种方法的作用：

void init（）：在该拦截器被初始化之后，在该拦截器执行拦截之前，系统回调该方法。对于每个拦截器而言，此方法只执行一次。

void destroy();该方法跟init()方法对应。在拦截器实例被销毁之前，系统将回调该方法

String intercept(ActionInvocation invocation) throws Exception;该方法是用户需要实现的拦截动作。该方法会返回一个字符串作为逻辑视图。

Intercept(ActionInvocation inv）方法返回值的含义：返回一个字符串作为逻辑视图，返回的值会到struts.xml中进行寻找，然后返回到一个jsp中。

1. **自定义拦截器的配置步骤、注意事项、配置文件中相关标签的作用。**

配置步骤：1.创建一个类，间接或直接的实现com.opensymphony.xwork2.

interceptor.Interceptor，定义其中的三个方法2.在struts.xml中进行拦截器的注册与配置

注意事项：1.继承AbstractInterceptor类时只需要实现intercept方法，init（）和destroy（），原因在于该类提供了init（）和destroy（）方法的空实现。2.intercept方法返回在字符串一定要在strut.xml中找到相应的逻辑视图。

配置文件中相关标签的作用3.需要注意的是：系统默认的拦截器栈应要放在前面，在加入自定义拦截器

拦截器的定义：<interceptor name=”拦截器名” class=”拦截器类”>

拦截器栈的定义：<interceptor-stack name=”拦截器名”>

<interceptor-ref name=”拦截器x”/>

<interceptor-ref name=”拦截器y”/>

</interceptor-stack>

全局拦截器：<default-interceptor-ref name="permissionStack"/>

1. **defaultStack的定义，该拦截器栈中包含的主要拦截功能。**

defaultStack的定义

<interceptor-stack name="defaultStack">

<interceptor-ref name="exception"/>

<interceptor-ref name="alias"/>

<interceptor-ref name="servletConfig"/>

<interceptor-ref name="i18n"/>

<interceptor-ref name="prepare"/>

<interceptor-ref name="chain"/>

<interceptor-ref name="debugging"/>

<interceptor-ref name="scopedModelDriven"/>

<interceptor-ref name="modelDriven"/>

<interceptor-ref name="fileUpload"/>

<interceptor-ref name="checkbox"/>

<interceptor-ref name="multiselect"/>

<interceptor-ref name="staticParams"/>

<interceptor-ref name="actionMappingParams"/>

<interceptor-ref name="params">

<param name="excludeParams">dojo\..\*,^struts\..\*</param>

</interceptor-ref>

<interceptor-ref name="conversionError"/>

<interceptor-ref name="validation">

<param name="excludeMethods">input,back,cancel,browse</param>

</interceptor-ref>

<interceptor-ref name="workflow">

<param name="excludeMethods">input,back,cancel,browse</param>

</interceptor-ref>

</interceptor-stack>

主要拦截功能：

1. params：解析HTTP请求参数，并设置成Action属性
2. validation：执行xxxAction-validate中定义的校验器
3. workflow:调用Action中的validate方法
4. i18n：支持国际化
5. createSession:创建HttpSession对象
6. timer:负责输出Action执行时间，分析性能
7. autowiring：自动装配，访问Spring中的Bean
8. **自定义过滤器的实现方法和配置步骤，拦截器与过滤器的对比**

配置步骤：1.创建一个类，继承Filter并实现doFilter方法 2.在web.xml文件中进行配置

实现方法：1.创建的类继承Filter并实现doFilter方法

拦截器和过滤器的对比：

1. 拦截器是基于java的反射机制的，而过滤器是基于函数回调。

2. 拦截器不依赖与servlet容器，过滤器依赖与servlet容器。

3. 拦截器只能对action请求起作用，而过滤器则可以对几乎所有的请求起作用。

4. 拦截器可以访问action上下文、值栈里的对象，而过滤器不能访问。

5. 在action的生命周期中，拦截器可以多次被调用，而过滤器只能在容器初始

化时被调用一次。

6.过滤器是在java web中，你传入的request,response提前过滤掉一些信息，或者提前设置一些参数，然后再传入servlet或者struts的 action进行业务逻辑。拦截器可通过的是符合条件的action。拦截器本身是一个普通的Java对象，它能动态拦截Action调用，

Action执行前后执行拦截器本身提供的各种个样的Web项目需求。也可以阻止Action的执行，同时也可以提取Action中可以复用的部分。

1. **Struts2的工作原理和基本工作过程**

工作原理：

Struts2框架的基本思想是采用MVC设计模式，即将应用设计成模型(Model)、视图(View)和控制器(Control)三个部分。

控制部分由DispatchFilter承担，模型部分是由一系列的拦截器(Interceptors)、Action组件和ActionContext组件组成，视图组件则是由Result组件组成；

基本工作过程：

  1：客户端在浏览器中输入一个url地址；

  2：这个url请求通过http协议发送给tomcat；

  3：tomcat根据url找到对应项目里面的web.xml文件；

  4：在web.xml里面会发现有struts2的配置；

  5：然后会找到struts2对应的struts.xml配置文件；

  6：根据url解析struts.xml配置文件就会找到对应的class；

  7：调用完class返回一个结果result，根据struts.xml返回到对应的jsp；

1. **问题及解决方案或思考**

思考：

1. 基础实验照着实验步骤一步一步往下写，没遇到什么困难。
2. 在书写代码时还未仔细思考过滤器和拦截器的作用，在进行实验总结时才明白了他的用法及作用，以及两者之间的区别。
3. **实验收获及总结**

收获及总结：

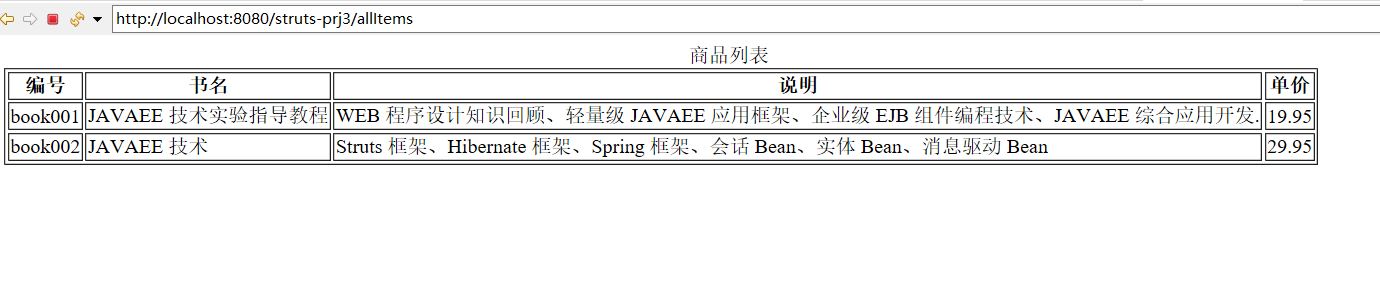
1. 清楚了拦截器、过滤器的配置方法、作用及两者之间的区别。
2. 对struts2中的工作原理和工作过程有了新的理解。

**二、提高实验——值栈与 OGNL**

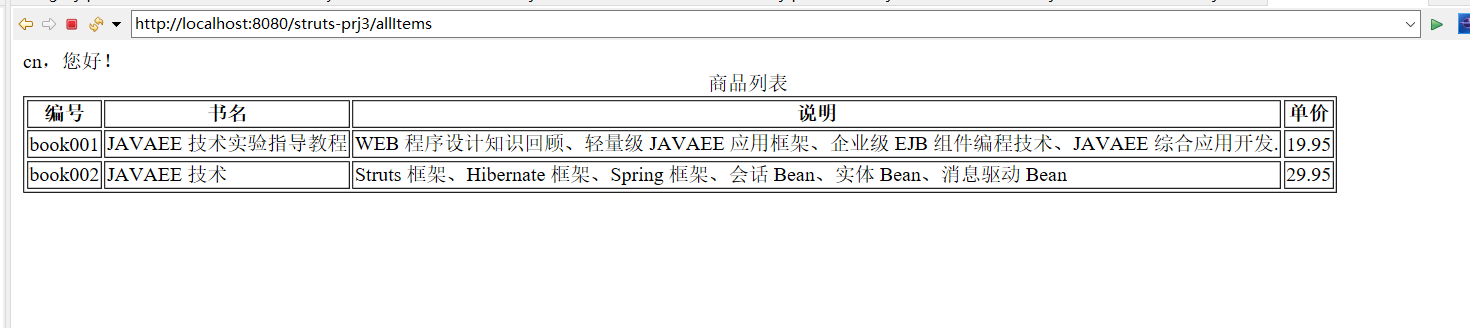
**（1）运行结果截图**

**1.提高实验第2小题**

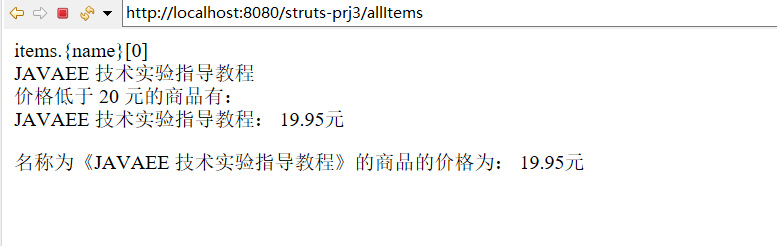
**登录后查看所有商品信息**



**2.提高实验第5小题**



**3.提高实验第8小题**



1. **ValueStack**

Struts API中的ValueStack称为值栈。值栈是一个数据区域，该区域中保存了应用范围内的所有数据和Action处理的用户请求数据。值栈被存储在ActionContext对象中，因此可以在任何节点访问其中的内容。一个valueStack对应一个action，valueStack贯穿整个action的生命周期。

ValueStack接口中主要方法:findValue()、setValue()。

开发步骤：1.通过request对象来获取valueStack 2.调用findValue（）函数获取对象 3.输出

1. **OGNL 可访问的对象和基本语法**

有访问权限的字段，无需getter方法，Ognl找不到getter方法会去直接访问字段；对于没有访问权限的字段则不能直接访问。

OGNL,即对象图导航语言，是Struts默认的表达式语言。当Action属性不是简单类型而是某个对象，如数组、集合时，Struts2利用OGNL表达式语言来访问ActionContext中的数据（Action数据放入 “区域”中，JSP将数据取出）。

**（4）整理 OGNL ,三种常用符号：#、%和$的 作用和使用方法**

1．#符号的三种用法

1）访问非根对象属性，例如示例中的#session.msg表达式，由于Struts 2中值栈被视为根对象，所以访问其他非根对象时，需要加#前缀。实际上，#相当于ActionContext. getContext()；#session.msg表达式相当于ActionContext.getContext().getSession(). getAttribute(“msg”) 。

2）用于过滤和投影（projecting）集合，如示例中的persons.{?#this.age>20}。

1. 用来构造Map，例如示例中的#{‘foo1’:’bar1’, ‘foo2’:’bar2’}。

2.%符号

1)%符号的用途是在标志的属性为字符串类型时，计算OGNL表达式的值。

1. $符号
2. 在国际化资源文件中，引用OGNL表达式。例:

reg.agerange=国际化资源信息：年龄必须在${min}同${max}之间。

2） 在Struts 2框架的配置文件中引用OGNL表达式.

**（5）问题及解决方案及思考**

1.查阅资料之后明白了OGNL相关语法和常用的三种符号的用法。

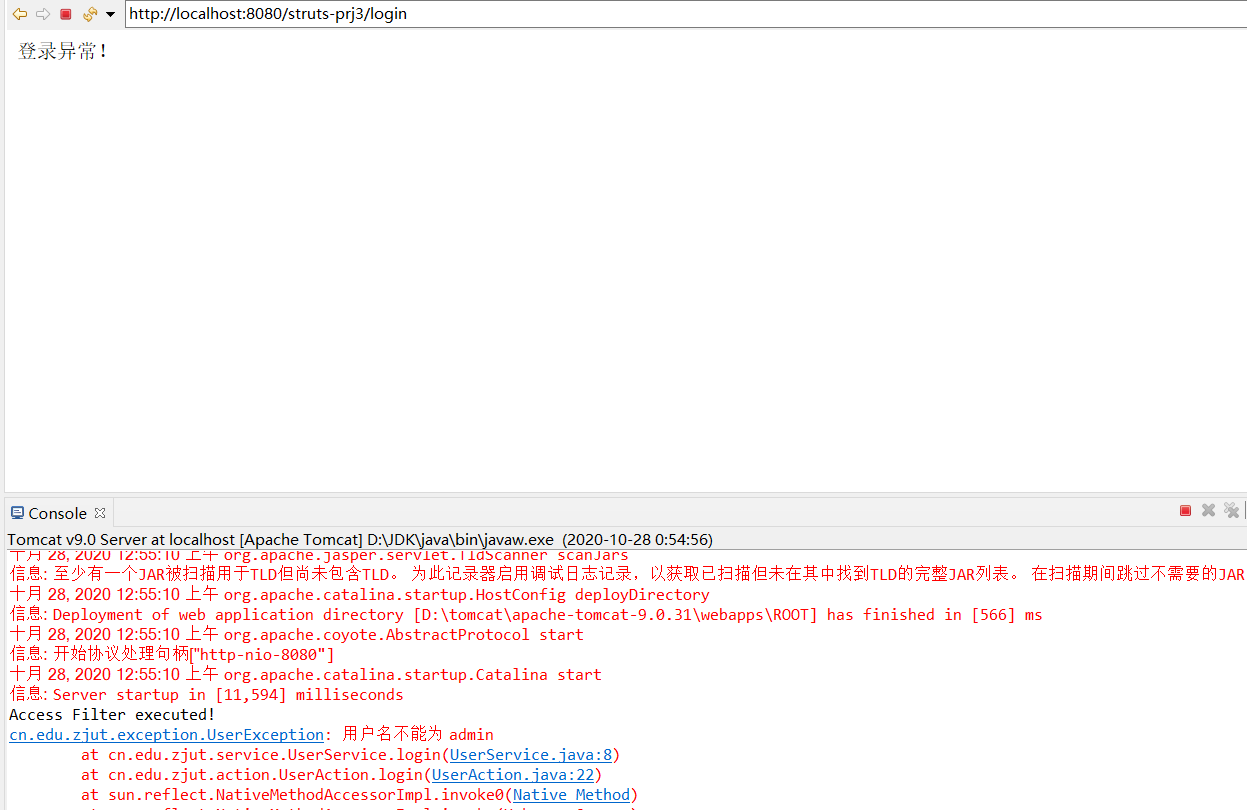
2.对ValueStack接口的实现与运用在查阅资料之后得以理解。

1. **收获及总结**
2. 对OGNL有了了解认识。
3. 对值栈的用法有了体会。

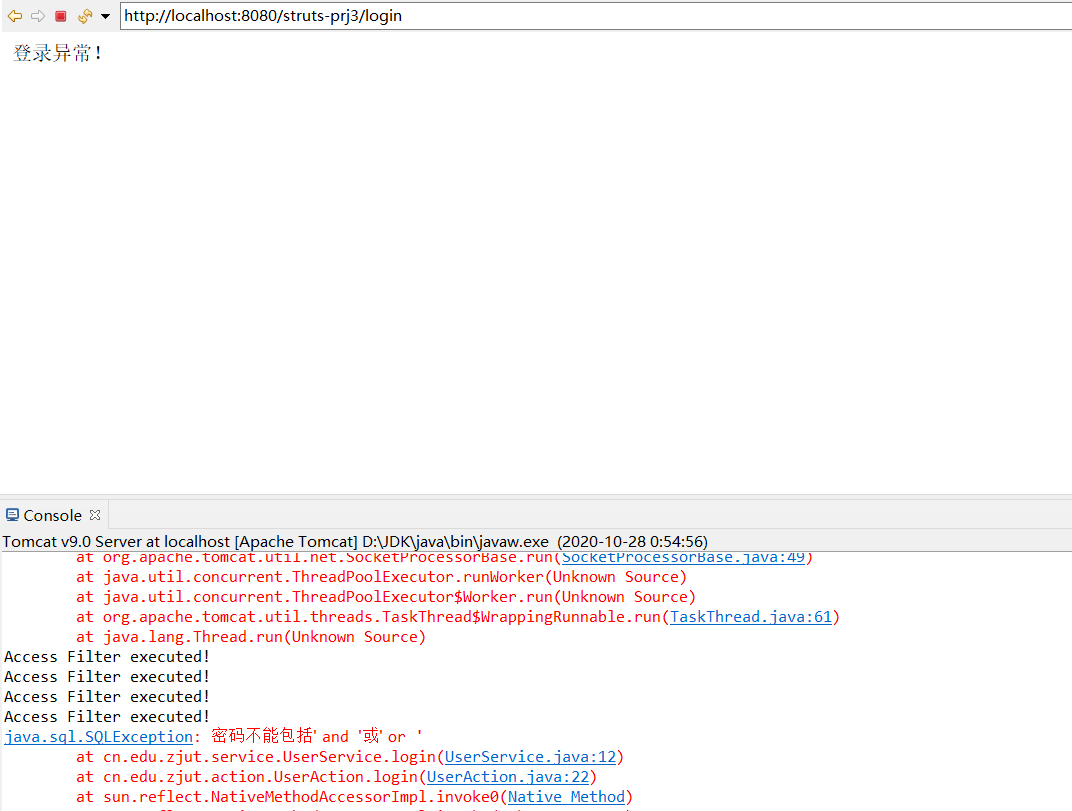
**三、扩展实验——Struts2 的异常处理**

**（1）运行结果截图；**

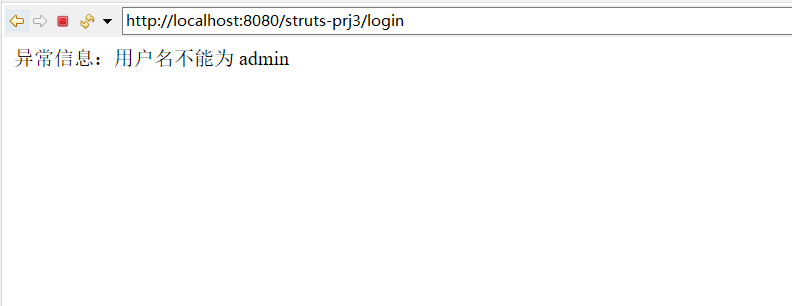
**1.用户名为admin**

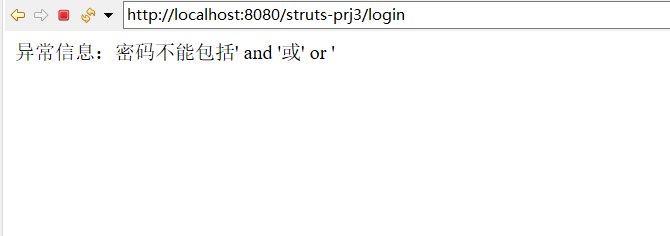


**2.密码中存在and或or**



**3.用Struts2框架处理异常**





1. **根据实验步骤 1-6，查找相关资料，整理自定义异常类的方法和步骤，并记录下来；**

自定义异常类的步骤：

1. 创建一个类继承Exception
2. 创建一个页面用于异常的视图
3. 在UserService类中书写产生异常的条件并抛出异常
4. 在UserAction中抓住异常并返回一个字符串
5. 配置struts.xml
6. **根据实验步骤 1-6，将 Action 使用 try/catch 捕获异常并返回结果视图的**关键代码和相关配置记录下来；

关键代码：

try {

if (userServ.login(loginUser)) {

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

return "success";

} else {

return "fail";

}

}catch (Exception e) {

e.printStackTrace();

return "exception";

}

相关配置：

<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>

<result name="exception">/loginException.jsp</result>

</action>

1. **根据实验步骤 7-10，查找相关资料，整理 Struts2 框架处理异常的机制，整理 struts.xml 文件配置异常映射的方法以及相关标签的作用，并记录下来；**
2. struts2进行异常处理首先需要添加exception拦截器，而默认拦截器栈已经加入了这个拦截器，所以不用特意的声明。
3. 当Action抛出异常时,会在局部中和全局中查找与之相匹配的异常,如果局部和全局中都有满足的异常映射,以局部的为准。
4. Struts2提供了一种声明式的异常处理方式。Struts2允许通过struts.xml文件来配置异常的处理。

配置异常映射：

1、局部异常映射：将<exception-mapping .../>元素作为<action .../>元素的子元素配置

2、全局异常映射：将<exception-mapping .../>元素作为<global-exception-mappings>元素的子元素配置

**（5）碰到的问题及解决方案或思考；**

1.直接粘贴pdf中的代码，发现密码中存在and也能够登录成功

解决方案：仔细查看UserException中的代码，发现他的包含的and或or前后是要带空格的，在尝试输入（空格）and（空格）后能够正常运行异常的代码。

2.出现以下类似报错



解决方案:仔细检查啊struts.xml的配置与UserAction中的写法，两者之中肯定至少有一个有错误。

**（6）实验收获及总结**

1.了解并清楚了异常的书写，尝试了异常的配置与显示。

2.了解了Struts2框架处理异常的方式，对异常处理有了更进一步的认识。