



JavaEE 实验三补充实验

实验报告

姓　　名　　 Mcrivers

班　　级　　 软件工程2102班

学　　号

提交日期　　 2023.11.12

目录

[一、基础实验——拦截器与过滤器 1](#_Toc151336870)

[1. 运行结果截图 1](#_Toc151336871)

[2.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 2](#_Toc151336872)

[3. 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 3](#_Toc151336873)

[4. 在 Struts2 核心包 struts2-core-\*.\*.\*.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 3](#_Toc151336874)

[5. 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 4](#_Toc151336875)

[6. 总结 Struts2 的工作原理和基本工作过程 5](#_Toc151336876)

[7. 碰到的问题及解决方案或思考 6](#_Toc151336877)

[8. 实验收获及总结 6](#_Toc151336878)

[二、提高实验——值栈与OGNL 7](#_Toc151336879)

[1. 运行结果截图 7](#_Toc151336880)

[2. 整理 ValueStack 接口及其主要方法的作用和开发步骤 8](#_Toc151336881)

[3. 整理 OGNL 可访问的对象和基本语法 8](#_Toc151336882)

[4. 整理 OGNL 三种常用符号：#、%和$的作用和使用方法 9](#_Toc151336883)

[5. 碰到的问题及解决方案或思考 9](#_Toc151336884)

[6. 实验收获及总结 9](#_Toc151336885)

[三、扩展实验——Struts2 的异常处理 10](#_Toc151336886)

[1. 运行结果截图 10](#_Toc151336887)

[2. 整理自定义异常类的方法和步骤 10](#_Toc151336888)

[3. 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 11](#_Toc151336889)

[4. 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 12](#_Toc151336890)

[5. 碰到的问题及解决方案或思考 13](#_Toc151336891)

[6. 实验收获及总结 13](#_Toc151336892)

# 一、基础实验——拦截器与过滤器

## 1. 运行结果截图

**13、将 struts-prj2 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**

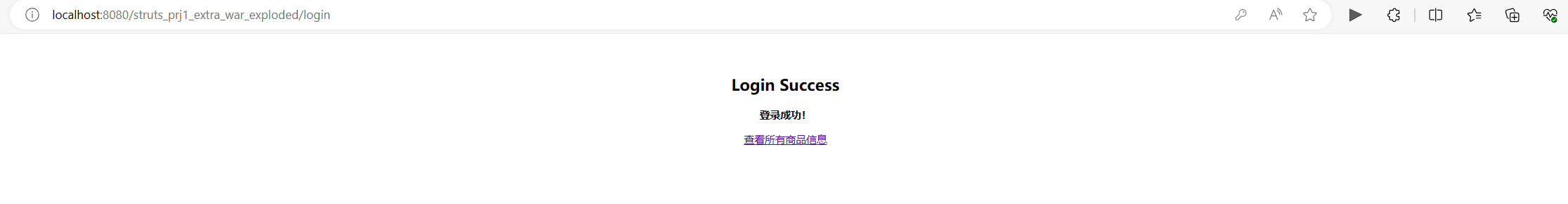


图1.1 登录成功界面



图1.2 商品列表界面

**17、重新将 struts-prj2 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**

观察到当直接访问loginSuccess.jsp页面点击超链接查看信息时，会返回到login.jsp页面要求登录；在login.jsp页面登录后进入loginSuccess.jsp后再点击超链接查看信息，则可以正常查看书单

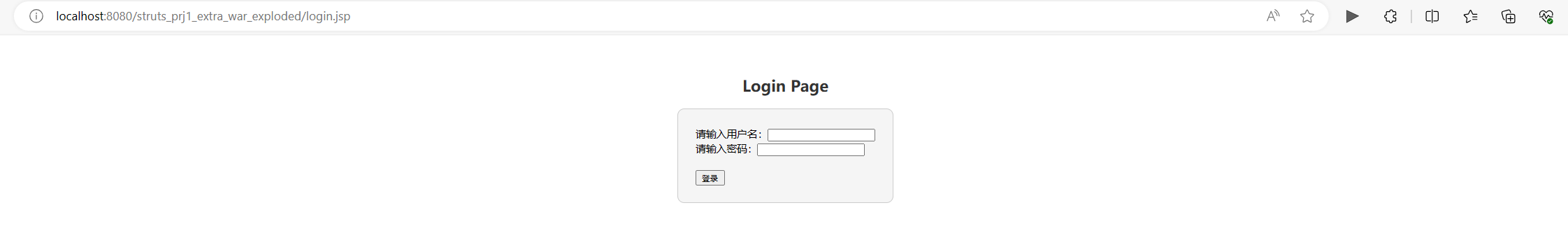


图1.3 当直接访问loginSuccess.jsp点击超链接时，会返回到login.jsp页面

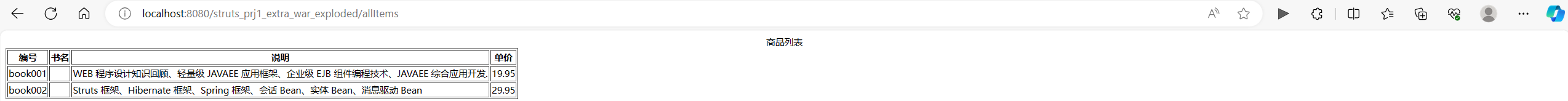


图1.4 当先访问login.jsp进行登录操作再点击超链接时，会正确显示书单

**20、重新将 struts-prj2 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。**

观察到，当直接尝试访问loginSuccess.jsp页面和itemList.jsp页面时，项目会出现404报错；而当登录后再尝试查看上述页面时，两个页面都可以正常访问



图1.4 直接尝试访问loginSuccess.jsp



图1.4 直接尝试访问itemList.jsp

## 2.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

**AuthorityInterceptor拦截器类：**

**作用：**

该拦截器用于实现用户权限控制功能，确保只有登录用户才有查看所有商品信息的权限。

**实现方法：**

**intercept(ActionInvocation invocation) 方法**：

作用：在这个方法中，进行拦截器的主要逻辑。在你的例子中，判断用户是否登录，如果登录了就执行invocation.invoke()，表示继续执行拦截器链，否则返回Action.LOGIN。

返回值：如果返回Action.LOGIN，表示拦截器终止，不继续执行后续的拦截器和Action，直接返回登录结果。

**init() 方法：**

作用：初始化拦截器，在拦截器被加载时执行。在你的例子中未提供实现。

**destroy() 方法：**

作用：销毁拦截器，在拦截器被卸载时执行。在你的例子中未提供实现。

**intercept(ActionInvocation inv) 方法返回值的含义：**

如果用户已登录，return invocation.invoke(); 表示继续执行拦截器链，允许用户访问被拦截的Action。如果用户未登录，return Action.LOGIN; 表示拦截器终止，不继续执行后续的拦截器和Action，并直接返回登录结果，即跳转到登录页面。

## 3. 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

**配置步骤：**

①创建自定义拦截器类

②实现拦截器类的实例

③配置拦截器

④将拦截器与特定的拦截点关联

**注意事项：**

①注意顺序

②避免死循环

③注意拦截器的生命周期

④注意拦截器的作用范围

**配置文件中相关标签的作用：**

<interceptor>标签：用于定义拦截器，包括拦截器的名称和拦截器类。

<interceptor-stack>标签：用于定义拦截器栈，将一组拦截器按顺序组织起来。

<default-interceptor-ref>标签：用于指定默认的拦截器栈。

<action>标签中的<interceptor-ref>标签：用于为特定的Action指定拦截器。

## 4. 在 Struts2 核心包 struts2-core-\*.\*.\*.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

exception拦截器：处理异常，负责捕获和处理Action中抛出的异常。

servletConfig拦截器：初始化ServletActionContext，提供对Servlet的配置信息的访问。

prepare拦截器：负责准备Action的数据，通常用于数据预处理。

i18n拦截器：处理国际化（Internationalization）的相关功能，负责加载对应的资源文件。

chain拦截器：负责调用拦截器栈中的下一个拦截器，实现拦截器的链式调用。

debugging拦截器：用于开启或关闭Struts2的调试模式，提供更详细的调试信息。

scopedModelDriven拦截器：如果使用了ModelDriven接口，此拦截器将确保ModelDriven对象与Action的生命周期一致。

params拦截器：处理请求参数，将参数设置到Action的属性中。

conversionError拦截器：处理类型转换错误，当请求参数无法转换为目标类型时，会执行此拦截器。

validation拦截器：处理验证逻辑，验证请求参数是否符合指定的规则。

workflow拦截器：实现了整个请求处理流程的控制，包括Action的选择和调用等。

## 5. 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

**自定义过滤器的实现方法和配置步骤：**

**实现方法：**

创建过滤器类： 编写一个类，实现javax.servlet.Filter接口。

实现过滤逻辑： 在过滤器类中实现doFilter方法，该方法包含过滤器的具体逻辑。

配置过滤器： 在web.xml文件中配置过滤器的映射和顺序。

**配置步骤：**

①创建过滤器类

②实现过滤逻辑

③配置过滤器

**拦截器与过滤器的比较：**

**共同点：**

位置： 都位于请求处理的不同阶段，用于对请求进行干预和处理。

灵活性： 可以通过配置顺序形成链式调用，实现多个拦截器或过滤器的协同工作。

**特点及区别：**

拦截器（Interceptor）：

Struts2特有： 主要用于Struts2框架中，用于对Action的调用进行拦截和处理。

与Action绑定： 拦截器与具体的Action类绑定，可以在Action的生命周期内执行预处理和后处理。

配置在Struts2配置文件中： 配置在struts.xml等Struts2配置文件中。

面向Action： 更关注业务逻辑，可以直接访问Action的上下文和数据。

过滤器（Filter）：

Servlet规范： 是Java Servlet规范的一部分，可以在任何Java Web应用中使用。

与Servlet容器绑定： 与具体的Servlet容器（如Tomcat）绑定，可以在请求进入Servlet容器前或离开容器后进行处理。

配置在web.xml中： 配置在web.xml文件中，独立于具体框架。

面向HTTP请求： 更关注与HTTP请求和响应相关的操作，对Servlet容器提供的请求和响应对象进行处理。

## 6. 总结 Struts2 的工作原理和基本工作过程

**Struts2的工作原理：**

客户端发起请求： 用户通过浏览器或其他客户端发起HTTP请求。

Servlet容器处理请求： 请求被Servlet容器（如Tomcat）接收和处理。

Filter拦截： 如果配置了Struts2的Filter，请求会被Struts2的StrutsPrepareAndExecuteFilter拦截。

创建ActionContext： Struts2创建一个ActionContext对象，该对象封装了HTTP请求、响应和其他相关信息。

创建ActionInvocation： Struts2创建一个ActionInvocation对象，用于管理Action的执行。

拦截器栈执行： 请求通过配置的拦截器栈，每个拦截器都有机会在Action执行前后进行处理。

执行Action： 最终调用相关的Action的方法执行业务逻辑。

结果处理： Action执行后，结果会被渲染，并交给结果视图进行显示。

响应生成： 生成最终的HTTP响应，返回给客户端。

**Struts2的基本工作过程：**

配置Action： 在Struts2的配置文件（通常是struts.xml）中配置Action的映射关系和拦截器栈。

接收请求： 当有HTTP请求进入时，Struts2的Filter拦截请求，创建ActionContext和ActionInvocation。

拦截器处理： 请求通过配置的拦截器栈，每个拦截器按照顺序执行，可以在Action执行前后进行一些处理，例如权限验证、日志记录等。

Action执行： 执行与请求对应的Action类中的方法，执行业务逻辑。

结果处理： 根据Action的执行结果，选择对应的结果视图进行渲染，生成最终的响应。

响应发送： 将生成的响应发送给客户端，完成请求-处理-响应的周期。

## 7. 碰到的问题及解决方案或思考

问题：在加入自定义过滤器后，出现报错：The content of element type "package" must match "(result-types?,interceptors?,default-interceptor-ref?,default-action-ref?,default-class-ref?,global-results?,global-allowed-methods?,global-exception-mappings?,action\*)".

分析：在<package>元素内部，各个子元素的排列是有顺序的

解决方案：调整子元素排列顺序，调整后的代码如下：

<package name="strutsBean" extends="struts-default" namespace="/">  
 <interceptors>  
 <interceptor name="authority"  
 class="cn.edu.zjut.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>  
 </action>  
 <action name="allItems" class="cn.edu.zjut.action.ItemAction"  
 method="getAllItems">  
 <result name="login">/login.jsp</result>  
 <result name="success">/itemList.jsp</result>  
 <interceptor-ref name="defaultStack"/>  
 <interceptor-ref name="authority"/>  
 </action>  
</package>

## 8. 实验收获及总结

在这个实验中，掌握了自定义拦截器的基本开发步骤和配置方法、自定义过滤器的基本开发步骤和配置方法，深入理解了拦截器和过滤器的特点和区别、默认拦截器栈 defaultStack 的主要拦截功能以及Struts2 的工作原理和基本工作过程。这让我对框架的工作原理和基本工作过程有了更清晰的认识，这将有助于我更好地利用 Struts2 进行Web应用的开发。

# 二、提高实验——值栈与OGNL

## 1. 运行结果截图

**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**

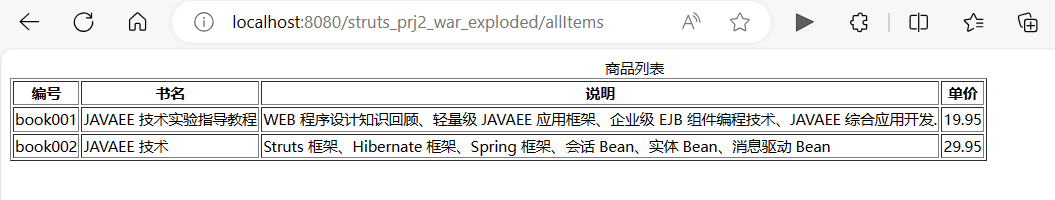


图2.1 商品列表

**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**

观察到左上角出现了“admin，您好”字样（admin是用户名）

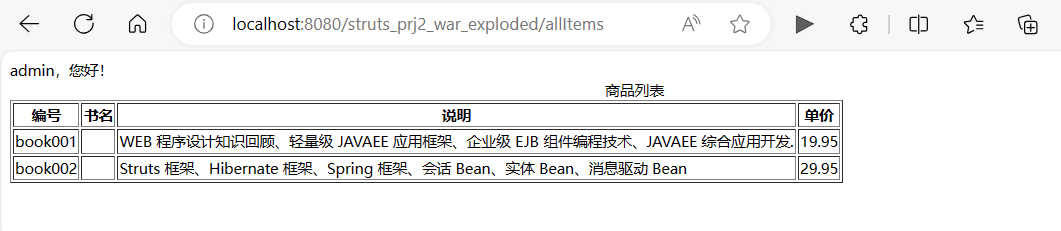


图2.2 商品列表

**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**

观察到，页面正确显示了价格小于 20 的商品和名称 为“JAVAEE 技术实验指导教程”的商品。同时，不用%的写法无法正确显示书名，但是使用%的写法可以正确显示书名

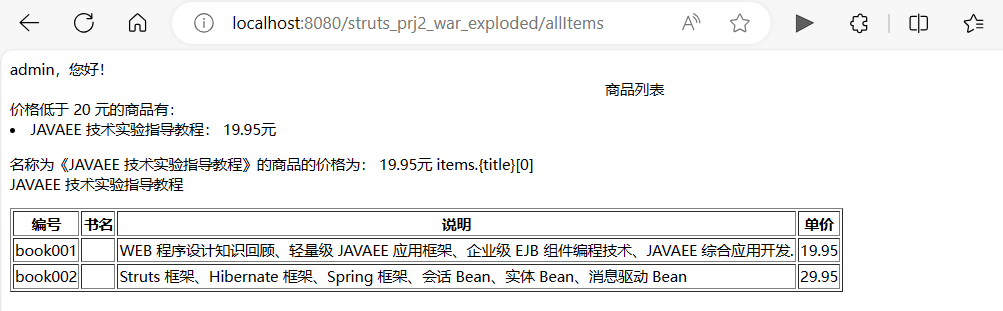


图2.3 商品信息

## 2. 整理 ValueStack 接口及其主要方法的作用和开发步骤

**ValueStack接口主要方法：**

findValue(String key): 根据指定的键（key）在值栈中查找对应的值。

set(String key, Object value): 将一个对象放入值栈并以指定的键存储。

push(Object o): 将对象推送到值栈的顶部。

getRoot(): 获取值栈的根对象。

getContext(): 获取与值栈相关联的上下文对象。

findString(String expr): 在值栈中查找并返回一个字符串值。

findObject(String expr): 在值栈中查找并返回一个对象。

findCreate(String expr, Class<?> toCreate): 查找或创建一个对象。

**开发步骤：**

存储数据到值栈： 在Action中通过set方法或push方法将数据存入值栈。

从值栈中获取数据： 在视图页面或拦截器等地方使用findValue、findString等方法获取值栈中的数据。

使用值栈的数据： 在页面中展示或处理值栈中的数据，例如展示在HTML页面上或传递给其他组件。

## 3. 整理 OGNL 可访问的对象和基本语法

**可访问的对象：**

①Action对象

②值栈对象

③JavaBean对象

**基本语法：**

属性访问： 使用点号（.）访问对象的属性。例如：person.name 将访问一个名为name的属性。

方法调用： 使用点号和括号调用对象的方法。例如：person.getName() 将调用getName()方法。

索引访问： 对于列表或数组，使用方括号（[ ]）访问特定索引位置的元素。例如：list[0] 获取列表中的第一个元素。

条件操作： 支持常规的条件操作符，如&&（与）、||（或）、!（非）等。

支持算术和关系运算符： 包括加减乘除、大于小于、等于不等于等运算符。

## 4. 整理 OGNL 三种常用符号：#、%和$的作用和使用方法

**1. #**

作用： # 符号用于引用值栈（Value Stack）中的对象。

使用方法：

访问Action类的属性或方法：#action.property 或 #action.method()

访问值栈中的对象属性或方法：#someObject.property 或 #someObject.method()

通常用于获取值栈中的对象或调用其方法。

**2. %**

作用： % 符号用于引用OGNL上下文中的特殊对象。

使用方法：

%{} 内部引用特殊对象：%{specialObject}

通常用法： 用于访问特殊的上下文对象，如Session、Request等。

**3. $**

作用： $ 符号用于引用OGNL表达式结果的临时变量。

使用方法：

$variableName：临时变量的名称。

通常用法： 用于存储和引用表达式计算结果的临时变量。

## 5. 碰到的问题及解决方案或思考

这个实验的过程很顺利，没有碰到任何问题。

## 6. 实验收获及总结

实验中，通过值栈及OGNL的使用，我深入理解了Struts2的核心机制。借助值栈，能高效地获取数据，并通过OGNL直接访问对象属性、调用方法。同时，我通过实践掌握了利用OGNL过滤集合、访问Session对象等操作，更深入地理解了Struts2框架的核心概念和机制。

# 三、扩展实验——Struts2 的异常处理

## 1. 运行结果截图

**6. 将 struts-prj2 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**

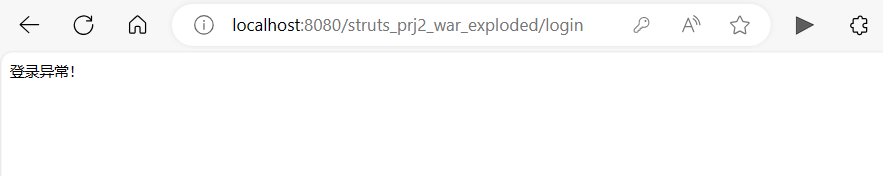
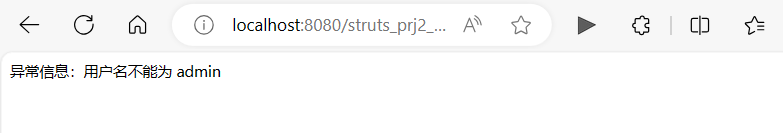


图3.1 登录异常界面

**10. 将 struts-prj2 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**

观察到，异常界面显示了异常原因

’ 图3.2 登录异常界面

## 2. 整理自定义异常类的方法和步骤

创建异常类： 创建一个新的类，让它扩展自Java提供的异常类，比如Exception或RuntimeException。

定义构造函数： 在自定义异常类中定义构造函数

添加异常信息： 在构造函数中自定义异常的信息，以便在捕获异常时提供有用的诊断信息。

处理异常：使用throw关键字抛出自定义异常对象。

## 3. 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：

public class UserAction extends ActionSupport {  
 public String login() {  
 UserService userServ = new UserService();  
 try {  
 if (userServ.login(loginUser)) {  
 // 登录成功逻辑  
 return "success";  
 } else {  
 // 登录失败逻辑  
 return "fail";  
 }  
 } catch (UserException ue) {  
 // 用户异常处理  
 ue.printStackTrace();  
 return "exception";  
 } catch (java.sql.SQLException sqlEx) {  
 // SQL异常处理  
 sqlEx.printStackTrace();  
 return "sql\_exception";  
 } catch (Exception e) {  
 // 其他异常处理  
 e.printStackTrace();  
 return "other\_exception";  
 }  
 }  
}

相关配置：

<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>  
<result name="sql\_exception">/sqlException.jsp</result>  
<result name="other\_exception">/otherException.jsp</result>  
</action>  
<!-- other actions -->  
</package>  
</struts>

## 4. 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

**Struts2 框架处理异常的机制：**

异常处理器：

Struts2提供了全局异常处理器，可以在struts.xml文件中配置。该处理器能够捕获并处理在Action中未被捕获的异常。

异常映射：

Struts2允许在struts.xml中设置异常映射，将特定的异常映射到处理逻辑或特定的结果页面。这样做可以使程序在出现特定异常时执行指定的操作，比如跳转到错误页面或者执行一些特殊的逻辑处理。

**struts.xml 文件配置异常映射的方法以及相关标签的作用：**

方法：

①定义全局异常处理器

②关联结果页面

③配置异常处理器在Action中使用

相关标签的作用：

<global-exception-mappings>:

作用： 定义全局的异常处理器，用于捕获Action中未被捕获的异常。

<exception-mapping>:

作用： 映射特定异常到处理器或结果页面。

属性：

exception：指定异常类或异常类的子类。

result：指定异常发生时要跳转的结果页面。

result-type：指定结果页面的类型（例如：redirectAction, dispatcher）。

statusCode：指定HTTP状态码。

<result>:

作用： 定义异常发生时跳转的结果页面。

属性：

name：定义结果页面的名称。

type：定义结果页面的类型。

## 5. 碰到的问题及解决方案或思考

实验过程很顺利，没有碰到任何问题。

## 6. 实验收获及总结

在这个扩展实验中，我深入了解了Struts2中处理异常的方式。通过配置异常映射和在Action中捕获异常，我学会了处理自定义异常，并将其映射到合适的视图页面。我也学习了如何在页面上输出异常信息，提高了对Struts2异常处理机制的理解。