（可选）实验三补充实验——Struts2 的工作流程

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-prj2，并将 Struts2 中的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 loginSuccess.jsp 页面，作为登录成功的视图，代码略；

3、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，代码略；

4、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，代码略；

5、在 struts-prj2 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，代码略；

6、在 struts-prj2 的 cn.edu.zjut.bean 中创建 Item.java，用于记录商品信息，代码略；

7、在 struts-prj2 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，部分代码如下：

```
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-prj2 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，部分代码如下：

```
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-prj2 中创建 itemList.jsp 页面，作为显示所有商品信息的视图，部分代码如下：

```
<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，部分代码如下：

```
<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，部分代码如下：

```
<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-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；

14、在 struts-prj2 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-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；

18、在 struts-prj2 中新建 cn.edu.zjut.filters 包，并在其中创建 AccessFilter.java 过滤器，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，部分代码如下：

```
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-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。

（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-*.*.*.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-prj2 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-prj2 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-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；

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-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。

（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。