

序号: 24

学号: 16430121



# 常州大学

## 实 验 报 告

课 程 名 称: Java EE 程序设计与应用开发

实 验 题 目: Struts 框架的使用

实 验 次 数: 第 四 次实验      实验时间: 2018 年 4 月 23 日

学 生 姓 名: 林锦雄

学      院: 信息数理 专 业 班 级: 计算机 162

指 导 教 师: 陆洁茹 评 阅 成 绩: \_\_\_\_\_

## 一、实验目的：

1. 掌握 Struts 标签库的使用方法。
2. 使用 Struts 标签库实现常用 Web 项目的关键页面。

## 二、实验仪器、设备

### 1、硬件环境

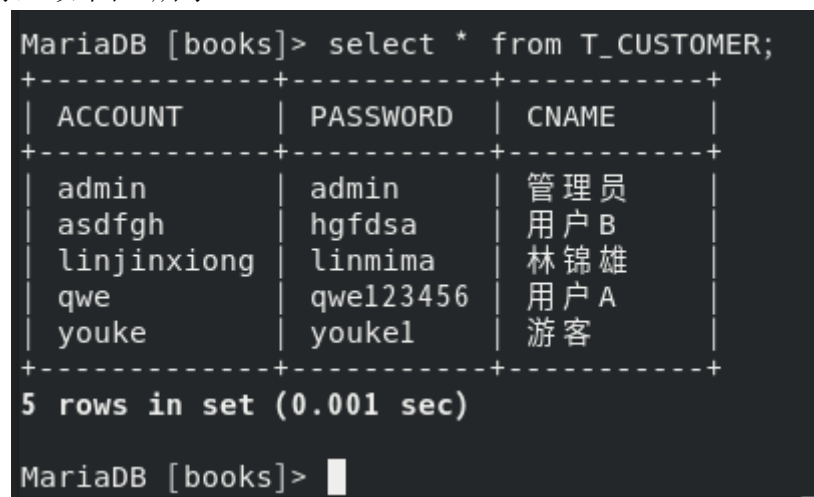
PC 微机；2G 以上内存；VGA 显示格式。

### 2、软件环境

Windows XP 以上操作系统，JDK，Tomcat 服务器等。

## 三、实验内容与要求

在数据库中建立表格 T\_CUSTOMER(ACCOUNT,PASSWORD,CNAME)，插入一些记录。如图 1 所示。



```
MariaDB [books]> select * from T_CUSTOMER;
```

ACCOUNT	PASSWORD	CNAME
admin	admin	管理员
asdfgh	hgfdsa	用户B
linjinxiong	linmima	林锦雄
qwe	qwel23456	用户A
youke	youkel	游客

```
5 rows in set (0.001 sec)
```

```
MariaDB [books]>
```

图 1. books 数据库中的 T\_CUSTOMER 表

1. 编写注册页面，使用 html 标签库中的标签，输入账号、密码、确认密码、姓名，完成注册的功能，并能够判断是否重复注册。如果重复注册，提示错误信息。

register.jsp 是用 html 标签库中的各种标签编写的注册页面，新用户通过输入账号、密码、确认密码和姓名信息提交到后台，让后台程序完成注册。该注册页面初始页面顶部是“欢迎注册账号!”的欢迎语，下方是注册表单，表单内容包括账号、密码、确认密码和姓名的输入、注册和取消按钮。如图 2 所示。

```
1. <!-- register.jsp -->
2. <h2>欢迎注册账号!</h2>
3. <html:form action="/register" method="post">
4.     <bean:message key="info.input" arg0="账号"></bean:message>
```

```

5. <html:text property="account"></html:text>
6. <html:errors property="account"></html:errors><br>
7. <bean:message key="info.input" arg0="密码"></bean:message>
8. <html:password property="password"></html:password>
9. <html:errors property="password"></html:errors><br>
10. <bean:message key="info.input" arg0="确认密码"></bean:message>
11. <html:password property="confirmPassword"></html:password>
12. <html:errors property="confirmPassword"></html:errors>
13. <html:errors property="notSamePassword"></html:errors><br>
14. <bean:message key="info.input" arg0="姓名"></bean:message>
15. <html:text property="name"></html:text>
16. <html:errors property="name"></html:errors><br><br>
17. <html:submit value="注册"></html:submit>
18. <html:cancel value="取消"></html:cancel><br>
19. <html:errors property="register"></html:errors>
20. </html:form>

```



图 2. 初始注册页面

RegisterForm.java 类是用来容纳前端网页表单里面的数据，具有和前端网页表单同名属性，继承 org.apache.struts.ActionForm，通过在 struts-config.xml 中注册，可以让系统认知它。

如下代码是其中的前端错误处理函数 validate 方法，它返回一个 ActionErrors 对象，是一个错误集合，专门用来容纳 ActionMessage，当内部有 ActionMessage 时，认为发生了前端错误。该方法进行前端错误验证的流程为：判断是否发生前端错误，如果发生前端错误，则把错误封装在 ActionMessage 对象中，然后放在 ActionErrors 对象中，返回。

在 RegisterForm 的 validate 方法中，能判断的前端错误有：表单元素是否为空、两次密码是否一致这两种前端错误。如图 3，图 4 所示。

```

1. //RegisterForm.java 类中的前端错误处理函数 validate
2. public ActionErrors validate(ActionMapping mapping, HttpServletRequest request) {
3.     ActionErrors errors = new ActionErrors();
4.     if(account.length() == 0) {
5.         ActionMessage message = new ActionMessage("error.null","账号");
6.         errors.add("account",message);
7.     }
8.     if(password.length() == 0) {

```

```

9.         ActionMessage message = new ActionMessage("error.null","密码");
10.         errors.add("password",message);
11.     }
12.     if(confirmPassword.length() == 0) {
13.         ActionMessage message = new ActionMessage("error.null","确认密码");
14.         errors.add("confirmPassword",message);
15.     } else if(!password.equals(confirmPassword)) {
16.         ActionMessage message = new ActionMessage("error.notSamePassword");
17.         errors.add("notSamePassword",message);
18.     }
19.     if(name.length() == 0) {
20.         ActionMessage message = new ActionMessage("error.null","姓名");
21.         errors.add("name",message);
22.     }
23.     return errors;
24. }

```



图 3. 表单元素为空



图 4. 两次密码不一致

DAO 层为专门的数据库访问层，AccountDao.java 类是检查账号信息的类。该类中的 AddAccount 方法传入前端表单的注册信息：账号、密码和姓名，连接数据库，查询需要注册的新账号是否在数据中已存在，如果已存在，返回 false，否则为新账号，把该新账号存入数据库中并返回 true。

```

1. //DAO 层 AccountDao.java 类，检查注册信息
2. public class AccountDao {
3.     private String Driver = "com.mysql.jdbc.Driver";
4.     private String URL="jdbc:mysql://localhost:3306/books?characterEncoding=UTF-8";
5.     private String user = "root";
6.     private String passwd = "bingjie123";
7.     public boolean AddAccount(String account,String password,String name) throws
Exception {
8.         Class.forName(Driver);
9.         Connection conn = DriverManager.getConnection(URL,user,passwd);
10.        String sql="select ACCOUNT from T_CUSTOMER where ACCOUNT='"+account+"'";
11.        Statement stat = conn.createStatement();
12.        ResultSet rs = stat.executeQuery(sql);
13.        if(rs.next()) {
14.            if(rs != null) rs.close();
15.            if(stat != null) stat.close();
16.            if(conn != null) conn.close();
17.            return false;

```

```

18.     }
19.     sql = "insert into T_CUSTOMER (ACCOUNT,PASSWORD,CNAME) value (?,?,?)";
20.     PreparedStatement ps = conn.prepareStatement(sql);
21.     ps.setString(1,account);
22.     ps.setString(2,password);
23.     ps.setString(3,name);
24.     ps.executeUpdate();
25.     if(ps != null) ps.close();
26.     if(rs != null) rs.close();
27.     if(stat != null) stat.close();
28.     if(conn != null) conn.close();
29.     return true;
30. }
31. }

```

Struts 框架中，要将 ActionForm 转交给 Action 来处理，Action 是负责业务逻辑的。RegisterAction 类继承 org.apache.struts.action.Action，重写 execute 方法来处理业务逻辑，mapping 参数的作用是访问配置文件，form 是 RegisterForm 传来的是封装好的前端注册表单数据。

RegisterAction 的 execute 方法通过实例化 AccountDao 并调用其 AddAccount 方法检查新注册的账号是否为新账号，如果返回 false，表示账号已被注册过，把错误信息封装在 ActionMessages 中，并把 ActionMessages 存入 request，然后通过 mapping 的 getInputForward 方法返回错误信息给前端页面。如图 5 所示。

如果新注册的账号确实为新账号，则 AddAccount 方法会把该新账号的各种信息存入数据库并返回 true，表示成功注册新账号，RegisterAction 的 execute 会调用 mapping 的 findForward 方法跳转到指定页面。“Success”的映射目标地址配置见 struts-config.xml。如图 6 所示。

```

1. //RegisterAction 类
2. public class RegisterAction extends Action {
3.     public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request, HttpServletResponse response) throws Exception {
4.         request.setCharacterEncoding("UTF-8");
5.         RegisterForm registerForm = (RegisterForm) form;
6.         String account = registerForm.getAccount();
7.         String password = registerForm.getPassword();
8.         String name = registerForm.getName();
9.         AccountDao accountDao = new AccountDao();
10.        boolean can = accountDao.AddAccount(account,password,name);
11.        if(!can){
12.            ActionMessages errors = new ActionMessages();
13.            ActionMessage message = new ActionMessage("error.account","注册",account + "已");
14.            errors.add("register",message);
15.            this.saveErrors(request,errors);
16.            return mapping.getInputForward();
17.        } else {
18.            return mapping.findForward("Success");
19.        }
20.    }
21. }

```

registerSuccess.jsp 为注册成功页面，该页面顶部是注册成功的提示语，下方是跳转至登录界面的按钮，点击按钮可以跳转至登录界面，然后使用新注册好的账号登录。如图 6 所示。

```
1. //registerSuccess.jsp
2. <h2>注册成功!请返回登陆!</h2><hr>
3. <form action="login.jsp">
4.     <input type="submit" value="登陆">
5. </form>
```



图 5. 重复注册

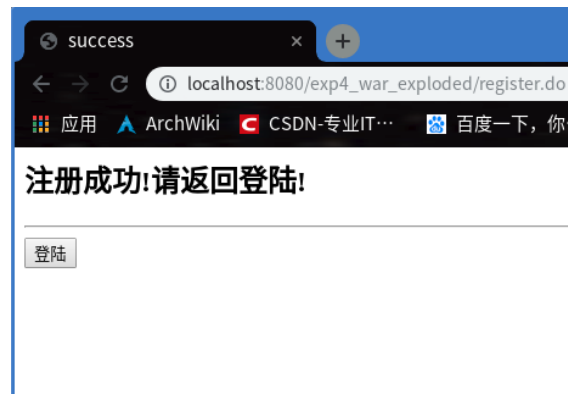


图 6. 注册成功

Struts 框架的配置文件，这里可以配置 ActionForm、Action、resources 等信息，文件之间的关联信息也需要在这里配置，如 ActionForm、Action 和前端网页的关联信息，RegisterForm、RegisterAction、register.jsp 的关联配置见 6~14 行的<action-mapping>。

```
1. <!-- struts-config.xml -->
2. <struts-config>
3.     <form-beans>
4.         <form-bean name="RegisterForm" type="Forms.RegisterForm"></form-bean>
5.     </form-beans>
6.     <action-mappings>
7.         <action path="/register"
8.             name="RegisterForm"
9.             type="Actions.RegisterAction"
10.            attribute="RegisterForm"
11.            input="/WEB-INF/jsp/register.jsp">
12.             <forward name="Success" path="/WEB-INF/jsp/registerSuccess.jsp"></forward>
13.         </action>
14.     </action-mappings>
15.     <message-resources parameter="ApplicationResources"></message-resources>
16. </struts-config>
```

ApplicationResources.properties 为 Struts 框架中的资源文件，在这里注册一系列 key-value 对，然后可以在前端页面通过<html:message>标签进行使用，大大减少了重复提示文字的书写，该资源文件还可以控制输出参数，也就是说通过它可以返回不同参数的各种信息，如不同的错误提示等待，使用该资源文件需要在

如上所示的 struts-config.xml 配置文件中配置，见上 15 行。

```
1. #ApplicationResources.properties 资源文件
2. info.input=请输入{0}:
3. error.null=<font color=red>{0}不能为空!</font>
4. error.length=<font color=red>{0}的长度必须在{1}~{2}位之间!</font>
5. error.type=<font color=red>{0}必须全部是数字!</font>
6. error.notSamePassword=<font color=red>两次密码输入不一致!</font>
7. error.account=<font color=red>{0}失败!账号已存在!</font>
```

至此，成功注册的新账号已通过 RegisterAction 添加到了指定的数据库表中。如图 7 所示。

```
MariaDB [books]> use books;
Database changed
MariaDB [books]> select * from T_CUSTOMER;
+-----+-----+-----+
| ACCOUNT | PASSWORD | CNAME |
+-----+-----+-----+
| 13525   | asd13525 | 湖畔阳光 |
| admin   | admin    | 管理员   |
| asdfgh  | hgfdsa   | 用户 B   |
| linjinxiong | linmima | 林锦雄   |
| qwe     | qwe123456 | 用户 A   |
| youke   | youkel   | 游客     |
+-----+-----+-----+
6 rows in set (0.001 sec)

MariaDB [books]> █
```

图 7. 成功注册新账号后的数据库

2. 编写登录页面，包含账号和密码两个表单元素。控制用户的输入，使用户输入的账号和密码必须不为空；账号必须在 5~8 位之间，密码必须在 6~10 位之间；账号必须全部是数字。要求所有的提示信息 and 错误信息都是从资源文件中得到并且是中文。

login.jsp 是用 html 标签库中的各种标签编写的登录页面，用户通过输入账号和密码提交到后台，让后台程序完成登录。该注册页面初始页面是一个登录表单，表单元素和各种提示信息错误信息都是从资源文件中得到，用户可以通过已注册过的输入账号和密码进行登录，然后是登录和取消按钮，再到下方则是“注册新账号”的链接，如果没有账号的用户可以点击该链接进入注册页面进行新账号的注册。如图 8 所示。

```
1. <!-- login.jsp -->
2. <html:form action="/login">
3.     <bean:message key="info.input" arg0="账号"></bean:message>
4.     <html:text property="account"/>
5.     <html:errors property="account"/> <br>
6.     <bean:message key="info.input" arg0="密码"></bean:message>
7.     <html:password property="password"/>
8.     <html:errors property="password"/> <br><br>
```



```

9.     <html:submit value="登陆"/> <html:cancel value="取消"/>
10.    <html:errors property="login"></html:errors>
11. </html:form>
12. <hr>
13. <a href="register.jsp">注册新账号</a>

```

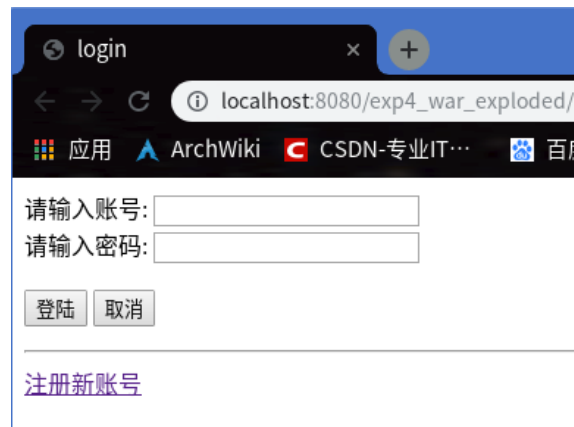


图 8. 初始登录页面

LoginForm.java 类是用来容纳前端网页登录表单里面的账号和密码，具有和前端网页登录表单同名属性，继承 org.apache.struts.ActionForm，也是通过在 struts-config.xml 中注册，可以让系统认知它。

在 LoginForm 的 validate 方法中，能判断的前端错误有：表单元素是否为空、账号是否全是数字以及账号密码的指定长度这三种前端错误。前两种前端错误如图 9 所示，后者如图 10 所示。

```

1. //LoginForm.java 类中的前端错误处理函数 validate
2. public ActionErrors validate(ActionMapping mapping, HttpServletRequest request) {
3.     ActionErrors errors = new ActionErrors();
4.     boolean tag = false;
5.     for(int i = 0; i < account.length(); i++) {
6.         if(account.charAt(i) < '0' || account.charAt(i) > '9') {
7.             tag = true;
8.         }
9.     }
10.    if(account.length() == 0) {
11.        ActionMessage message = new ActionMessage("error.null", "账号");
12.        errors.add("account", message);
13.    } else if(account.length() < 5 || account.length() > 8) {
14.        ActionMessage message = new ActionMessage("error.length", "账号", 5, 8);
15.        errors.add("account", message);
16.    } else if(tag == true) {
17.        ActionMessage message = new ActionMessage("error.type", "账号");
18.        errors.add("account", message);
19.    }
20.    if(password.length() == 0) {
21.        ActionMessage message = new ActionMessage("error.null", "密码");
22.        errors.add("password", message);
23.    } else if(password.length() < 6 || password.length() > 10) {
24.        ActionMessage message = new ActionMessage("error.length", "密码", 6, 10);
25.        errors.add("password", message);

```



```

26.     }
27.     return errors;
28. }

```

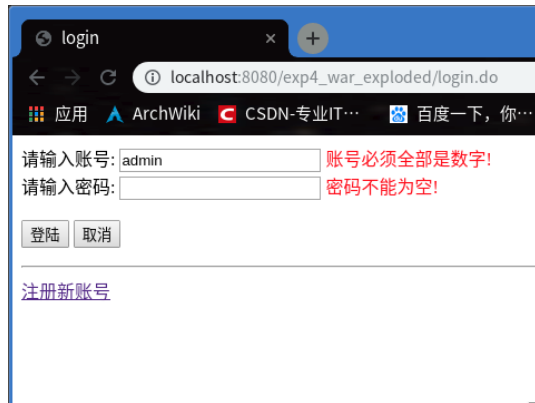


图 9. 登录信息为空或不符合

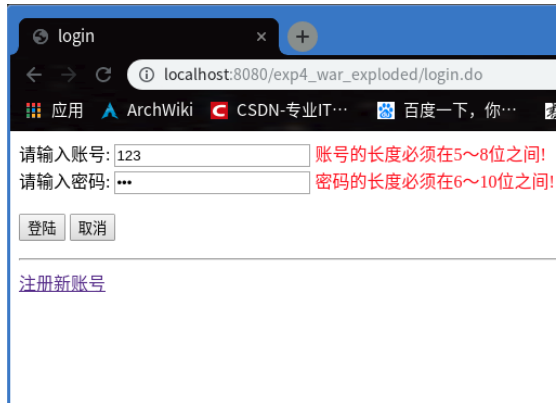


图 10. 登录账号密码长度不正确

DAO 层为专门的数据库访问层，AccountDao.java 类是检查账号信息的类。该类中的 queryAccount 方法传入前端表单的登录信息：账号和密码，连接数据库，查询正在登录的账号是否在数据中存在，如果存在，返回 true，否则该登录账号为新账号，并未注册，返回 false。

```

1. //位于 AccountDao.java 类中的登录账号检查方法
2. public boolean queryAccount(String account, String password) throws Exception{
3.     Class.forName(Driver);
4.     Connection conn = DriverManager.getConnection(URL,user,passwd);
5.     String sql = "select ACCOUNT from T_CUSTOMER where ACCOUNT=? and PASSWORD=?";
6.     PreparedStatement ps = conn.prepareStatement(sql);
7.     ps.setString(1,account);
8.     ps.setString(2,password);
9.     ResultSet rs = ps.executeQuery();
10.    if(rs.next()) {
11.        return true;
12.    } else {
13.        return false;
14.    }
15. }

```

LoginAction 的 execute 方法通过实例化 AccountDao 并调用其 queryAccount 方法检查正在登录的账号是否已注册过，如果正在登录的账号在数据库中查不到，返回 false，表示该账号还没被注册，execute 方法把错误信息封装在 ActionMessages 中，并把 ActionMessages 存入 request，然后通过 mapping 的 getInputForward 方法返回错误信息给前端页面。如图 11 所示。

如果已注册返回 true，表示账号已被注册过，允许登录，execute 方法通过调用 mapping 的 findForward 方法跳转到指定页面。“Success”的映射目标地址配置见 struts-config.xml。如图 12 所示。

```

1. //LoginAction.java 类中的 execute 方法
2. LoginForm loginForm = (LoginForm) form;
3. String account = loginForm.getAccount();

```

```

4. String password = loginForm.getPassword();
5. AccountDao accountDao = new AccountDao();
6. boolean can = accountDao.queryAccount(account,password);
7. if(!can){
8.     ActionMessages errors = new ActionMessages();
9.     ActionMessage message = new ActionMessage("error.login");
10.    errors.add("login",message);
11.    this.saveErrors(request,errors);
12.    return mapping.getInputForward();
13. } else {
14.    return mapping.findForward("Success");
15. }

```

loginSuccess.jsp 为注册成功页面，该页面只有一句话，是登录成功的提示语。如图 12 所示。

```

1. <!-- loginSuccess.jsp -->
2. <h2>登陆成功!欢迎使用!</h2>

```

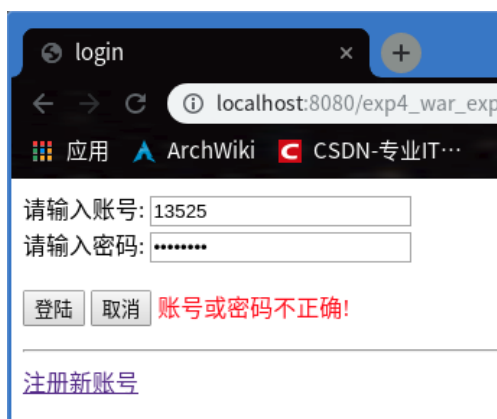


图 11. 登录账号或密码不正确

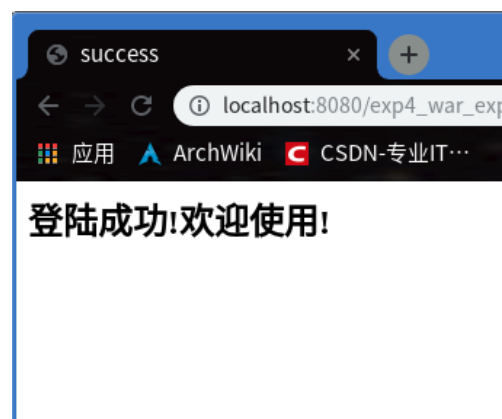


图 12. 登录成功页面

Struts 框架的配置文件，这里可以配置 ActionForm、Action、resources 等信息，文件之间的关联信息也需要在这里配置，如 ActionForm、Action 和前端网页的关联信息，LoginForm、LoginAction、login.jsp 的关联配置见 6~14 行的 <action-mapping>

```

1. <!-- struts-config.xml -->
2. <struts-config>
3.     <form-beans>
4.         <form-bean name="LoginForm" type="Forms.LoginForm"></form-bean>
5.     </form-beans>
6.     <action-mappings>
7.         <action path="/login"
8.             name="LoginForm"
9.             type="Actions.LoginAction"
10.            attribute="LoginForm"
11.            input="/WEB-INF/jsp/login.jsp">
12.             <forward name="Success" path="/WEB-INF/jsp/loginSuccess.jsp"></forward>
13.         </action>
14.     </action-mappings>
15.     <message-resources parameter="ApplicationResources"></message-resources>
16. </struts-config>

```

ApplicationResources.properties 为 Struts 框架中的资源文件，对比上面，需要新添加如下一行 key-value 值。

1. #ApplicationResources.properties 资源文件
2. `error.login=<font color=red>账号或密码不正确!</font>`

## 四、实验心得

本次实验，我掌握了 Struts 框架的基础理解和使用，通过搭建一个登录和注册系统，使用 Struts 标签库，Struts 资源文件，Struts 错误处理，Struts 框架的编写方式去编写这个系统。

Struts 框架是一个基于 MVC 设计模式的框架，MVC 思想的核心概念为：Model 封装应用程序的数据结构和事务逻辑，集中体现应用程序的状态，当数据状态改变的时候，能够在视图中体现出来，JavaBean 非常适合这个角色。View 是 Model 的外在表现，用 JSP 表示。Controller 是对用户的输入进行响应，将模型和视图联系到一起，负责将数据写到模型中，并调用视图，用 Java Servlet 来进行。

在 Struts 中工作流程表现为：前端 JSP 的数据提交给指定 ActionServlet，ActionServlet 读取配置文件，通过提交地址，找到 Action 对应的类，通过 name 属性找到对应的 ActionForm 类，将表单数据封装到 ActionForm 中，提交给 Action，最后 ActionServlet 调用 Action 的 execute 方法处理业务逻辑，处理完后返回一个 ActionForward 对象给 ActionServlet，ActionServlet 跳转到指定的结果页面。

搭建这个登录和注册的系统，让我大致地了解并学会操作 Struts 框架，并且进行了基础地应用来加深对这个框架的理解。为今后学校 SSH 主流框架打下基础。