序号: 24

学号: 16430121

常州大学

J2EE 程序设计大作业报告

题　　目：毕业设计题目查询系统

学 生 姓 名：林锦雄

学　院（系）：信息数理学院　专 业 班 级：计算机 162

指 导　教 师：陆洁茹　专业技术职务：讲师

完成时间：2019 年 5 月 7 日 ~ 2019 年 5 月 28 日

1、系统设计

1.1 方案的原理、特点与选择依据

（1）Struts2 是一个基于 MVC 设计模式的 Web 应用框架，它本质上相当于一个 servlet，在 MVC 设计模式中，Struts2 作为控制器(Controller)来建立模型与视图的数据交互。Struts 2 是 Struts 的下一代产品，是在 struts 1 和 WebWork 的技术基础上进行合并的全新的 Struts 2 框架。其全新的 Struts 2 的体系结构与 Struts 1 的体系结构差别巨大。Struts 2 以 WebWork 为核心，采用拦截器的机制来处理用户的请求，这样的设计也使得业务逻辑控制器能够与 ServletAPI 完全脱离开，所以 Struts 2 可以理解为 WebWork 的更新产品。

Strut2 框架具有八个特点：

a) 实现了 MVC 模式，层次结构清晰，使程序员只需关注业务逻辑的实现
b) 丰富的标签库，大大提高了开发的效率
c) Struts2 提供丰富的拦截器实现
d) 通过配置文件，就可以掌握整个系统各个部分之间的关系
e) 异常处理机制，在配置文件中配置异常的映射，可对异常做相应的处理
f) Struts2 的可扩展性高
g) 面向切面编程的思想在 Strut2 中也有了很好的体现
h) 支持 I18N（国际化）

（2）Spring 是一个轻量级控制反转(IoC)和面向切面(AOP)的容器框架。Spring 是一个开源框架，是为了解决企业应用程序开发复杂性而创建的。Spring 使用基本的 JavaBean 来完成以前只可能由 EJB 完成的事情。然而，Spring 的用途不仅限于服务器端的开发。从简单性、可测试性和松耦合的角度而言，任何 Java 应用都可以从 Spring 中受益。

Spring 框架具有七个特点：

a) 方便解耦，简化开发（高内聚低耦合）
b) AOP 编程的支持
c) 非侵入式设计
d) 支持声明式事务处理
e) 方便程序的测试
f) 方便集成各种优秀框架
g) 降低 Jave EE API 的使用难度

（3）Hibernate 是一个开放源代码的对象关系映射框架，它对 JDBC 进行了非常轻量级的对象封装，它将 POJO 与数据库表建立映射关系，是一个全自动的 orm 框架，hibernate 可以自动生成 SQL 语句，自动执行，使得 Java 程序员可以随心所欲的使用对象编程思维来操纵数据库。 Hibernate 可以应用在任何使用

JDBC 的场合，既可以在 Java 的客户端程序使用，也可以在 Servlet/JSP 的 Web 应用中使用，最具革命意义的是，Hibernate 可以在应用 EJB 的 J2EE 架构中取代 CMP，完成数据持久化的重任。

Hibernate 框架具有七个特点：

a) 对象/关系数据库映射(Basic O/R Mapping)
b) 透明持久化(Persistent)
c) 支持事务 Transaction (org.Hibernate.Transaction)
d) 开发效率高
e) 移植性很好
f) 缓存机制。提供一级缓存和二级缓存
g) 简洁的 HQL 编程

（4）SSH 为 struts+spring+hibernate 的一个集成框架，是目前较流行的一种 Web 应用程序开源框架。集成 SSH 框架的系统从职责上分为四层：表示层、业务逻辑层、数据持久层和域模块层（实体层），以帮助开发人员在短期内搭建结构清晰、可复用性好、维护方便的 Web 应用程序。其中使用 Struts 作为系统的整体基础架构，负责 MVC 的分离，在 Struts 框架的模型部分，控制业务跳转，利用 Hibernate 框架对持久层提供支持，Spring 做管理，管理 struts 和 hibernate。

SSH 框架的优势：

a) 典型的三层构架体现 MVC（模型 Model,视图 View 和控制）思想，可以让开发人员减轻重新建立解决复杂问题方案的负担和精力。便于敏捷开发出新的需求，降低开发时间成本。

b) 良好的可扩展性，ssh 主流技术有强大的用户社区支持它，所以该框架扩展性非常强，针对特殊应用时具有良好的可插拔性，避免大部分因技术问题不能实现的功能。

c) 良好的可维护性，业务系统经常会有新需求，三层构架因为逻辑层和展现层的合理分离，可使需求修改的风险降低到最低。随着新技术的流行或系统的老化，系统可能需要重构，ssh 构架重构成功率要比其他构架高很多。

d) 优秀的解耦性，很少有软件产品的需求从一开始就完全是固定的。客户对软件需求，是随着软件开发过程的深入，不断明晰起来的。因此，常常遇到软件开发到一定程度时，由于客户对软件需求发生了变化，使得软件的实现不得不随之改变。ssh 三层构架，控制层依赖于业务逻辑层，但绝不与任何具体的业务逻辑组件耦合，只与接口耦合；同样，业务逻辑层依赖于 DAO 层，也不会与任何具体的 DAO 组件耦合，而是面向接口编程。采用这种方式的软件实现，即使软件的部分发生改变，其他部分也不会改变。

1.2 系统的需求分析

（1）功能分析

毕业设计题目查询系统主要分为 4 个模块：

a) 用户注册和登录模块

b) 毕业设计题目所属学院录入

c) 毕业设计题目录入

d) 毕业设计题目信息查询（支持模糊查询）。

管理用户注册和登录，毕业设计题目的录入，修改和查询功能。系统功能示意图，如图 1 所示。

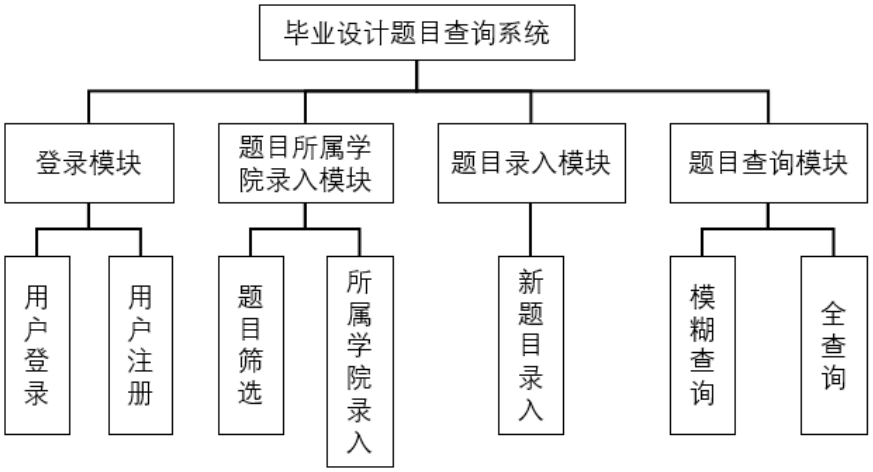

图 1. 系统功能示意图

（2）用户需求分析：

a) 用户需要进行登录才能使用系统，登录需要预先注册好账号才有登录该系统的权限。

b) 进入系统后有录入新毕业设计题目、修改毕业设计题目以及查询毕业设计题目等功能。

（3）系统功能需求分析：

a) 用户登录：用户在登录过程中出现不合法的操作，系统会给出相应的提示，如账号和密码不能为空，密码可能不正确以及账号不存在等不合法操作。

b) 用户注册：如果账号不存在，则需要用户进入账号注册页面进行新账号的注册，注册账号过程中也会给予用户相应的操作提示，如输入不能为空，账号可能已被注册等情况。

c) 毕业设计题目所属学院录入：进入该功能页面，应首先展示数据库中的全部毕业设计题目，每道题目都有其负责人、题目名称和所属学院，前两项都不允许被修改，第三项所属学院允许被更新，即在录入新的毕业设计题目操作时可以暂不录入该题目的所属学院。另外考虑数据库中的毕业设计题目数据较大时，提供一个关键字筛选题目的功能内嵌在其中，用户可以通过关键字筛选出负责人姓名或毕业设计题目名称或所属学院名中含有关键字的匹配项，然后再对需要更

新所属学院的题目进行更新。

d) 毕业设计题目录入：在该功能页面中，提供一个含有一个毕业设计题目所有属性的空表单，用户在表单中录入新题目的属性，然后点击录入，即可把新题目录入到系统中，当然，系统会保证不会重复录入的情况。

e) 毕业设计题目信息查询：进入该功能页面，应该有一个搜索表单，用户可以通过键入关键字然后查询数据库中毕业设计题目负责人、题目名称或学院名称中包含关键字的匹配项返回显示给用户。

f) 用户退出：在系统任意界面都提供注销用户的退出系统按钮。注销用户后，即返回到登录界面，用户需要重新登录才能再次进入系统，否则通过 URL 直接访问系统内部页面都会被检测到并转至登录页面等到用户重新登录系统。

1.2 系统详细设计

（1）数据库设计：

在开发毕业设计题目查询网页系统时，根据系统的各模块实现的功能，在数据库中抽象出两个实体类，分别为用户实体和毕业设计题目实体。其中用户实体有账号、密码和用户名三个属性，账号为主键。毕业设计题目实体有题目负责人、题目名称和所属学院三个属性，题目负责人为主键。

根据两个实体的属性，在数据库 Graduation 中设计两张对应的表，分别为用户表和毕业设计题目表，其中用户表中，账号为主键，不能为空；毕业设计题目表中，题目负责人为主键，不能为空，题目名称也不能为空，题目所属学院允许为空，可以在后期录入。两张表的属性表，如表 1、表 2 所示。

表 1 用户属性表

属性	类型	是否为空	自动生成	描述
ACCOUNT	varchar	Not null		主键
PASSWORD	varchar			账号密码
STUDENT	varchar			用户名

表 2 毕业设计题目属性表

属性	类型	是否为空	自动生成	描述
STUDENT	varchar	Not null		主键
TOPIC	varchar	Not null		题目名称
COLLEGE	varchar			所属学院

（2）系统软件设计：

● 用户登录和注册模块：用户使用系统的初始界面就是登录界面，在该界面有一个由账号和密码输入框以及登录按钮组成的表单，用户通过输入自己的账号和密码进行登录以进入系统。登录过程中会出现 4 种情况：

a) 账号或密码为空，账号或密码都不能为空，如果任意为空系统则在对应的位置打印红色提示用户输入不能为空。

b) 账号不存在，用户输入的账号在数据库中检索不到，表明该账号不存在，有可能是用户输入错误，也有可能是用户并未注册该系统的账号。系统返回账号不存在的红色提示。

c) 密码错误，这个情况表示账号在数据库中可以找到，但用户登录时输入的密码和数据库中对应账号的密码不匹配，所以系统返回红色提示用户登录密码输入错误的信息。

d) 账号密码都正确，成功登录并跳转至系统内部，用户可以开始使用该系统。

登录、注册模块的详细程序流程图，如图 2 所示。

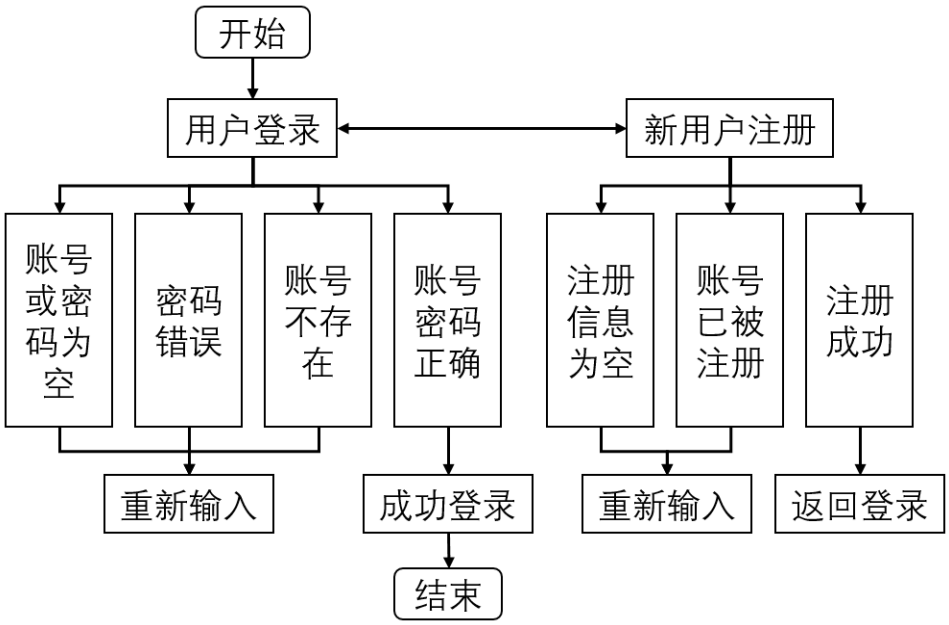

图 2. 登录、注册模块程序流程图

● 毕业设计题目所属学院录入模块：用户在系统中进入并使用毕业设计题目所属学院录入功能。该界面有一个筛选毕业设计题目的筛选框，和一个展示毕业设计题目的表格。

a) 录入所属学院：用户可以在表格中选中想要更新题目所属学院的题目，然后在其对应的所属学院输入框中键入所属学院信息，点击更新按钮即可把数据更新到数据库中，如果想要更新的数据和数据库中相同，系统返回重复录入的红

色提示。

b) 筛选毕业设计题目：考虑到数据库中的毕业设计题目数据较大的情况，用户可以使用毕业设计题目筛选功能，用户可以通过在筛选框中键入关键字筛选出负责人姓名或毕业设计题目名称或所属学院名中含有关键字的匹配项，然后刷新下方的毕业设计题目展示表格，显示更精准更符合用户需要的毕业设计题目表。然后再对需要更新所属学院的题目进行所属学院录入更新操作。

毕业设计题目所属学院录入模块的详细程序流程图，如图 3 所示。

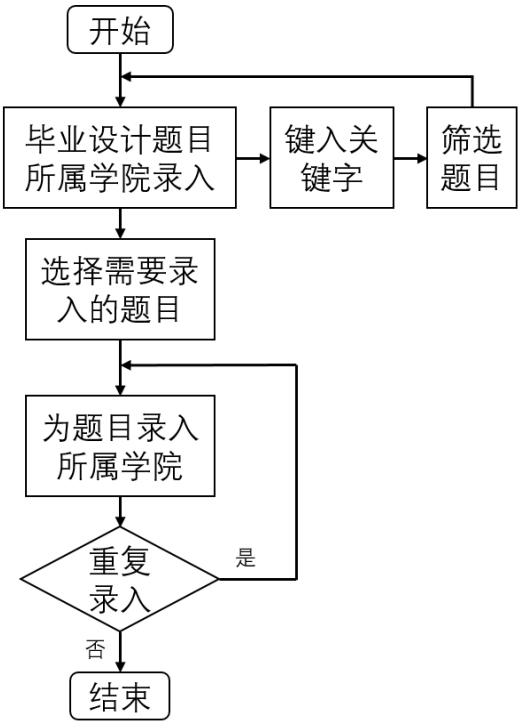

图 3. 毕业设计题目所属学院录入模块程序流程图

- 其他模块省略。

2、系统实现

2.1 用户登录模块

（1）Hibernate 框架

首先建立数据库层，即 DAO。

User.java 是数据库 USER 表对应的持久化类，有三个属性，分别和 USER 表中的三个字段对应：account 对应 USER 表中的 ACCOUNT 字段，表示该用户的账号；password 属性映射到 USER 表的 PASSWORD 字段，表示该用户的账号密码；student 属性映射到 USER 表中的 STUDENT 字段，表示该用户的用户名。具体实体类设计如下所示。

```java
//User.java
@Entity
@Table(name = "USER", schema = "Graduation")
public class User {
    private String account;
    private String password;
    private String student;
    //省略 getter 和 setter 方法
    //省略 equals 和 hashCode 重载方法
}
```

在 Graduation 数据库中创建 USER 表，并插入一些数据，数据内容如图 4 所示。

```
MariaDB [(none)]> use Graduation
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [Graduation]> select * from USER;
+----------+----------+-----------+
| ACCOUNT  | PASSWORD | STUDENT   |
+----------+----------+-----------+
| 16430121 | 123456   | 林锦雄    |
| aaa      | qwe      | 气温      |
| yuchen   | yuchen   | 于晨      |
+----------+----------+-----------+
3 rows in set (0.002 sec)
```

图 4. 创建 USER 表并写入一些数据

Table.hbm.xml 是 Hibernate 的实体类映射文件，在该文件中书写了数据库 Graduation 中的 USER 表和 User.java 实体类的映射关系：account 属性映射到 USER 表的 ACCOUNT 字段，为主键；password 属性映射到 USER 表的 PASSWORD 字段；student 属性映射到 USER 表中的 STUDENT 字段。

```xml
<!-- Table.hbm.xml -->
<hibernate-mapping>
    <class name="Table.User" table="USER" schema="Graduation">
        <id name="account" column="ACCOUNT"/>
        <property name="password" column="PASSWORD"/>
        <property name="student" column="STUDENT"/>
    </class>
</hibernate-mapping>
```

hibernate.cfg.xml 是 Hibernate 的配置文件，在该文件中书写了数据库配置信息以及映射文件的注册信息，数据库的连接后期转交 Spring 框架处理，所以这里我已经把数据库配置信息转交到了 Spring 框架的配置文件中。

```xml
<!-- hibernate.cfg.xml -->
<hibernate-configuration>
  <session-factory>
```

```
    <property name="dialect">org.hibernate.dialect.MySQLDialect</property>
    <mapping class="Table.User"/>
    <mapping resource="Table/Table.hbm.xml"/>
  </session-factory>
</hibernate-configuration>
```

LoginDao.java 是毕业设计题目系统登录模块的 DAO 类，该类有一个 Login 方法，传入登录账号和登录密码，然后用 sessionFatory 对象打开一个 session，并通过 User.java 数据库表映射实体类和传入的登录账号查询数据库 USER 表中匹配的用户。

user 对象为空，表示账号不存在，返回空；

user 对象不为空，但 user 对象中的密码和传参中的登录密码不一致，则密码不正确，把 user 对象的 password 属性置空串，返回 user 对象；

user 对象完全与登录信息吻合，最后提交事务并关闭 session，返回 user 对象。

```
//LoginDao.java
//登陆模块
public class LoginDao {

    private User user;  //一个 User 对象
    private SessionFactory sessionFactory;  //session 工厂对象
    //用 sessionFactory 打开一个 session,然后根据账号传参查询数据库
    //查询不到的匹配项,表示账号不存在,返回空
    //查询到匹配项,但密码与传参不一致,把得到的对象密码置空串返回
     //成功查询到匹配项并且密码正确,返回查询结果对象
     public User Login(String account, String password) {
         Session session = sessionFactory.openSession();
         user = session.get(User.class, account);
         session.close();
         if(user == null) {
             return null;
         } else if(!user.getPassword().equals(password)) {
             user.setPassword("");
         }
         return user;
     }
}
```

如果登录者发现其还没拥有本网站的账号，即需要注册新账号。RegisterDao.java 就是毕业设计题目系统注册模块的 DAO 类，该类有一个 Register 方法，传入一组注册账号的信息，包含账号、密码和用户名，然后用 sessionFatory 对象打开一个 session，并通过 User.java 数据库表映射实体类和传入的注册信息中的账号查询数据库 USER 表中是否已存在该账号。如果账号已存在，表示账号已被注册，关闭 session 对象并返回 false；如果注册信息是新的，则打开一个事务，新建一个 User 对象，并把注册信息写入该对象，然后通过 session 保存该对象，提交事务，即把新注册的账号信息写入数据库，最后关闭 session 对象并返

回 true。

```
//RegisterDao.java
//注册模块
public class RegisterDao {

    private User user;  //一个 User 对象
    private SessionFactory sessionFactory;  //session 工厂对象
    //用 sessionFactory 打开一个 session,然后根据账号传参查询数据库
    //查询不到匹配项,允许注册该新账号,开启一个事务并把新账号的数据写入数据库,返回 true
    //查询到匹配项,表示账号已存在,注册失败,返回 false
     public boolean Register(String account, String password, String student) {
         Session session = sessionFactory.openSession();
         user = session.get(User.class, account);
         if(user == null) {
             Transaction ts = session.beginTransaction();
             user = new User();
             user.setAccount(account);
             user.setPassword(password);
             user.setStudent(student);
             session.save(user);
             ts.commit();
             session.close();
             return true;
         } else {
             session.close();
             return false;
         }
     }
}
```

（2）Struts2 框架

利用 Struts2 框架搭建基本的 MVC 模型，即前端用 JSP 显示，然后提交表单到后端对应 Action 业务处理类，Action 类调用 DAO 层访问数据库并得到结果，然后处理结果，最后根据结果的不同跳转到不同的页面。

在 web.xml 中配置 Struts2 框架的过滤器，过滤目标是整个工程。

```
<!-- web.xml -->
<!-- struts2 过滤器 -->
<filter>
    <filter-name>struts2</filter-name>
    <filter-class>org.apache.struts2.dispatcher.filter.StrutsPrepareAndExecuteFil
ter</filter-class>
</filter>
<filter-mapping>
    <filter-name>struts2</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

login.jsp 是毕业设计题目所属学院登录模块的页面。其最上方是系统对使用者表示欢迎的语句。然后是一个登录表单，用户可以输入自己的账号和密码，点

击登录按钮进行登录。

登录信息不能为空，即账号和密码都不能为空，否则，点击登录按钮后会提示用户登录信息不能为空的相应红色提示语。

登录密码输入不正确也会反馈用户，提示用户重新输入正确的登录密码。

当如果用户使用没注册的账号进行登录，系统监测到，则提示用户账号未注册，请用户注册后使用。

最后是一个注册链接，提供给需要注册新账号的用户使用，点击链接可以跳转至注册页面。

如果未登录就非法访问系统内部页面，系统可以监测到，并返回到此登录页面，登录监测见后面部分介绍。

```
<!-- login.jsp -->
<h2>欢迎登陆毕业设计题目系统!</h2><hr>
<table>
<form action="login.action" method="post">
    <tr>
        <td>请输入账号：</td>
        <td><input type="text" name="account"></td>
        <td><s:fielderror fieldName="account" cssStyle="color:red"/></td>
    </tr>
     <tr>
         <td>请输入密码：</td>
         <td><input type="password" name="password"></td>
         <td><s:fielderror fieldName="password" cssStyle="color:red"/></td>
     </tr>
     <tr>
         <td><input type="submit" value="登陆" style="font-size: 15px"/></td>
         <td><a href="register.jsp" style="font-size: 15px">注册</a></td>
     </tr>
     <tr><s:actionerror cssStyle="color:red"/></tr>
</form>
</table><hr>
```

初始登录页面如图 5 所示，有欢迎登录的欢迎语句，包含需要输入登录账号、登录密码和登录按钮的登录表单，还有跳转到注册页面的链接。

图 5. 登录页面

register.jsp 是毕业设计题目所属学院注册模块的页面。其上方是欢迎注册的欢迎语。然后是一个注册信息填写的表单，用户填写需要注册的账号信息：账号、

密码和用户名后点击注册按钮进行注册。

注册信息均不能为空，即需要完整地填写账号、密码和用户名这些资料，如果其中有空信息，系统会反馈相应的红色的提示信息，提醒用户注册信息不能为空。

注册过程中，系统发现用户想要注册的账号在数据库中已存在，即已被注册，系统会取消注册，并反馈红色提示信息，提醒用户注册的账号已被注册。

如果注册的信息没问题，则把新账号写入数据库，并返回成功注册的提示语，并请用户跳转至登录页面进行登录操作。

```
<!-- register.jsp -->
<h2>欢迎注册毕业设计题目系统!</h2><hr>
<table>
<form action="register.action" method="post">
    <tr>
        <td>请输入账号：</td>
        <td><input type="text" name="account"></td>
        <td><s:fielderror fieldName="account" cssStyle="color:red"/></td>
    </tr>
     <tr>
         <td>请输入密码：</td>
         <td><input type="password" name="password"></td>
         <td><s:fielderror fieldName="password" cssStyle="color:red"/></td>
     </tr>
     <tr>
         <td>请输入姓名：</td>
         <td><input type="text" name="student"></td>
         <td><s:fielderror fieldName="student" cssStyle="color:red"  /></td>
     </tr>
     <tr>
         <td><input type="submit" value="注册" style="font-size: 15px"/></td>
         <td><a href="login.jsp" style="font-size: 15px">登陆</a></td>
     </tr>
     <tr><s:actionerror cssStyle="color:red"/>
     <s:actionmessage cssStyle="font-size: 20px"/></tr>
</form>
</table><hr>
```

初始注册页面如图 6 所示，有欢迎注册的欢迎语句，还有包含账号、密码、姓名这三个注册信息和注册按钮的注册表单，还有一个跳转到登录页面的链接。

图 6. 注册页面

struts.xml 是 Struts2 的配置文件，在该配置文件中注册毕业设计题目所属学院登录和注册模块中需要用到的两个 Action 类，LoginAction.java 和 RegisterAction.java，以及对他们进行一些跳转逻辑地址的映射。由于后期整合 Struts2 框架和 Spring 框架后，Action 类的生成转交 Spring 框架进行，所以这里的 class 为别名，与 Spring 框架中的相应配置信息对应。

```
<!-- struts.xml -->
<struts>
<package name="struts2" extends="struts-default">
    <action name="login" class="login" >
        <result name="success">/WEB-INF/JSP/graduation.jsp</result>
        <result name="input">/WEB-INF/JSP/login.jsp</result>
        <result name="error">/WEB-INF/JSP/login.jsp</result>
    </action>
    <action name="register" class="register">
         <result name="success">/WEB-INF/JSP/register.jsp</result>
         <result name="input">/WEB-INF/JSP/register.jsp</result>
         <result name="error">/WEB-INF/JSP/register.jsp</result>
     </action>
</package>
</struts>
```

LoginAction.java 是毕业设计题目所属学院登录模块中的处理类，该类中有登录账号、登录密码和登录 Dao 对象三个属性。其中的登录账号和登录密码对应前端的表单属性；登录 Dao 对象用于在数据库中检查登录信息是否正确。

在 execute 方法中，调用 loginDao 对象的 Login 方法，传入登录账号和密码，把结果返回给 user 对象。

user 为空，表示账号在数据库中找不到，即账号不存在，通过继承 ActionSupport 类并使用 addActionError 方法记录错误信息并显示到前端页面，提醒用户登录账号不存在。

user 的密码为空字符串，表示数据库中密码与用户输入密码不一致，即密码不正确，同样地可以使用 addActionError 方法记录错误信息后显示在前端页面，提示用户登录密码不正确。

如果登录账号和密码都正确，则开启 session，并把登录者的用户名保存在 session 中，然后返回，通过相应的逻辑地址进入系统。

在 LoginAction 中重载 validate 方法，validate 方法可以使用 addFieldError 方法把前端的输入错误信息保存并反馈给前端页面，例如输入内容不能为空或者输入内容的合法性。在这里重写两种错误情况，输入的登录账号和登录密码都不能为空的情况，如果为空则返回红色的提示信息。

```
//LoginAction.java
//登陆模块
public class LoginAction extends ActionSupport {

```

```
5.      private String account;      //登陆账号
6.      private String password;     //登陆密码
7.      private LoginDao loginDao;   //登陆 Dao
8.
9.      @Override
10.      public String execute() {
11.          User user = loginDao.Login(account, password);
12.          //user 为空,表示账号在数据库中找不到,即账号不存在
13.          //user 的密码为空字符串,表示数据库中密码与用户输入密码不一致,即密码不正确
14.          //登陆信息正确,在 session 中保存正在登陆账号的信息,进入系统
15.          if(user == null) {
16.              addActionError("账号不存在!");
17.              return "error";
18.          } else if(user.getPassword().equals("")) {
19.              addActionError("密码不正确!");
20.              return "error";
21.          } else{
22.              Map session = ActionContext.getContext().getSession();
23.              session.put("user",user);
24.              return "success";
25.          }
26.      }
27.
28.      @Override
29.      public void validate() {
30.          super.validate();
31.          //登陆信息中账号、密码不能为空,为空时返回错误显示
32.          if(account == null || account.equals("")) {
33.              this.addFieldError("account","账号不能为空");
34.          }
35.          if(password == null || password.equals("")) {
36.              this.addFieldError("password","密码不能为空");
37.          }
38.      }
39. }
```

在登录过程中，登录账号和密码均不能为空，其中之一为空都会反馈错误信息，并等待用户重新输入登录信息进行登录。详细错误反馈如图 7 所示。

图 7. 登录账号或密码为空

登录过程中，如果用户输入的登录密码不正确，系统在查询数据库确认错误后，反馈密码不正确的红色错误信息，如图 8 所示。

图 8. 登录密码不正确

登录过程中，用户使用的登录账号，系统在数据库中并不能成功检索到，则判断其使用的账号并未注册，反馈账号不存在的红色错误信息，如图 9 所示。

图 9. 登录账号不存在

登录账号和密码都正确，系统在数据库中能检索到与之对应的账号信息，则为成功登录，系统跳转至菜单使用页面。

最上面显示欢迎登录的用户使用该系统的信息，会自动匹配登录账号的用户名。然后是菜单，有三个项目：1、毕业设计题目所属学院录入 2、新毕业设计题目信息录入 3、毕业设计题目信息查询。使用者可以通过每个项目后边对应的选择按钮进入到对应的项目中。最后是一个退出登录的按钮，使用该按钮可以退出当前账号的登录状态，并跳转至登录页面。详细如图 10 所示。

图 10. 登录成功

RegisterAction.java 是毕业设计题目所属学院注册模块中的处理类，该类中有注册账号、注册密码、注册人用户名和注册 Dao 对象四个属性。其中的注册账号、注册密码和注册人用户名对应前端的表单属性；注册 Dao 对象用于在数据库中检查注册信息是否正确。

在 execute 方法中，调用 registerDao 对象的 Register 方法，传入注册账号、注册密码和注册人用户名，返回一个 boolean 值给 insert 变量，根据 insert 的值下选择对应的逻辑地址进行跳转。

insert 为真，表示注册成功，新账号已成功加入数据库，继承 ActionSupport 类，调用 addActionMessage 方法在前端显示注册成功的提示信息。

insert 为假，表示注册失败，正在注册的账号在数据库中已存在，即已被注册，调用 addActionError 方法在前端显示注册失败的红色字体提示语，提示用户账号已被注册，请输入正确的注册信息重新注册。

在 RegisterAction 中重载 validate 方法，validate 方法可以使用 addFieldError 方法把前端的输入错误信息保存并反馈给前端页面。在这里重写三种错误情况，输入的注册账号、注册密码和注册人用户名都不能为空的情况，如果为空则返回红色的提示信息。

```
1. //RegisterAction.java
2. //注册模块
3. public class RegisterAction extends ActionSupport {
4.
5.     private String account;          //注册账号
6.     private String password;         //注册密码
7.     private String student;          //注册人用户名
8.     private RegisterDao registerDao;//注册 Dao
9.
10.     @Override
11.     public String execute() {
```

```
12.         boolean insert = registerDao.Register(account,password,student);
13.         //insert 为真,表示注册成功,新账号已成功加入数据库
14.         //insert 为假,表示注册失败,正在注册的账号在数据库中已存在,即已被注册
15.         if(insert) {
16.             addActionMessage("注册成功!请返回登陆!");
17.             return "success";
18.         } else {
19.             addActionError("注册失败!账号已注册!");
20.             return "error";
21.         }
22.     }
23.
24.     @Override
25.     public void validate() {
26.         super.validate();
27.         //注册信息中账号、密码、姓名不能为空,为空时返回错误显示
28.         if(account == null || account.equals("")) {
29.             this.addFieldError("account","账号不能为空");
30.         }
31.         if(password == null || password.equals("")) {
32.             this.addFieldError("password","密码不能为空");
33.         }
34.         if(student == null || student.equals("")) {
35.             this.addFieldError("student","姓名不能为空");
36.         }
37.     }
38. }
```

在注册新账号的过程中，注册账号、注册密码以及注册人用户名均不能为空，其中之一为空都会反馈错误信息，并等待用户重新输入正确的注册信息进行注册。详细错误反馈信息如图 11 所示。

图 11. 注册账号、密码或用户名为空

在注册新账号的过程中，用户填写的注册账号信息中希望注册的账号，系统在数据库中能成功检索到已存在，则判断其希望注册的账号在此之前已被注册过，系统反馈注册失败，账号已被注册的红色错误信息，如图 12 所示。

图 12. 注册账号已被注册过

注册账号在数据库中并不存在，即是全新未使用的，允许注册该新账号，系统把新账号的全部信息作为一条用户数据写入数据库中，并返回注册成功，请用户返回登录页面进行登录系统的提示信息。如图 13 所示。

图 13. 注册成功

在成功注册之后，查询数据库可以发现，新的账号信息已经成功地写入到指定的数据库的用户表中。如图 14 所示。

```
MariaDB [Graduation]> select * from USER;
+----------+-----------+-----------+
| ACCOUNT  | PASSWORD  | STUDENT   |
+----------+-----------+-----------+
| 16416213 | chenqiang | 陈强      |
| 16430121 | 123456    | 林锦雄    |
| aaa      | qwe       | 气温      |
| yuchen   | yuchen    | 于晨      |
+----------+-----------+-----------+
4 rows in set (0.001 sec)
```

图 14. 注册成功后的数据库 USER 表数据

（3）Spring 框架

把 Hibernate 框架和 Spring 框架进行整合，把 Struts2 框架和 Spring 框架进行整合，由 Spring 框架控制 Hibernate 框架中对数据库的连接和映射，Struts2 框架

中的各种对象的生成也转交 Spring 处理，例如 Struts2 框架中的 Action 需要创建 Hibernate 框架中的数据库访问类即 DAO 层，这里将 DAO 层的对象创建转交给 Spring 框架完成，还有 Struts2 框架中各种类的装配也转交 Spring 进行处理，可以大大地降低程序的耦合度。

在 web.xml 中配置 Spring 框架的监听器，配置该监听器后，所有对象的装配都优先被 Spring 框架监测，如果可以在 Spring 框架中装配则在 Spring 框架中装配，否则再按照原方法装配。在这里还配置 Spring 框架配置文件 applicationContext.xml 的配置文件的路径。

```
<!-- web.xml -->
<!-- spring 监听器 -->
<listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>

<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath*:applicationContext.xml</param-value>
</context-param>
```

applicationContext.xml 是 Spring 框架的配置文件，用于整合 Hibernate 框架和 Struts2 框架，让 Hibernate 框架需要做的数据库配置信息和连接操作，实体类与数据库表的映射配置，实体类的生成都交由 Spring 框架装配；再整合 Struts2 框架，让 Struts2 框架中需要生成的各种 Action 以及需要创建的各种 Hibernate 框架中的 DAO 层对象都转交 Spring 框架装配。

首先是装配一个 Table.Design 实体类的 design 对象。然后装配 Struts2 的登录和模块的 Action 和 DAO 以及过滤题目模块的 Action 和 DAO，并为他们注入 sessionFactory 对象，为 Action 注入 DAO 层的对象。还有数据库连接的配置信息装配注入到 sessionFactory 中。这样就完成了 Spring 框架对 Hibernate 框架以及 Struts2 框架的整合，使得大部分的装配都在 Spring 框架中进行，类与类之间的依赖大大降低，程序之间的耦合度也大大降低。

```
<!-- applicationContext.xml -->
<!-- Bean -->
<bean name="user" class="Table.User" scope="prototype"></bean>
<bean name="design" class="Table.Design" scope="prototype"></bean>

<!-- 登陆模块 -->
<bean name="loginDao" class="DAO.LoginDao" scope="prototype">
    <property name="sessionFactory" ref="sessionFactory"></property>
</bean>
<bean name="login" class="Action.LoginAction" scope="prototype">
    <property name="loginDao" ref="loginDao"></property>
</bean>

<!-- 注册模块 -->
```

```
<bean name="registerDao" class="DAO.RegisterDao" scope="prototype">
    <property name="sessionFactory" ref="sessionFactory"></property>
</bean>
<bean name="register" class="Action.RegisterAction" scope="prototype">
    <property name="registerDao" ref="registerDao"></property>
</bean>

<!-- sessionFactory -->
<bean name="sessionFactory" class="org.springframework.orm.hibernate5.LocalSessionFactoryBean" scope="prototype">
    <property name="dataSource" ref="dataSource"></property>
    <property name="mappingLocations" value="classpath*:Table/Table.hbm.xml"></property>
    <property name="mappingResources" value="hibernate.cfg.xml"></property>
</bean>
<bean name="dataSource" class="org.apache.commons.dbcp2.BasicDataSource" scope="prototype">
    <property name="driverClassName" value="com.mysql.jdbc.Driver"></property>

    <property name="url" value="jdbc:mysql://localhost:3306/Graduation?characterEncoding=UTF-8"></property>
    <property name="username" value="root"></property>
    <property name="password" value="bingjie123"></property>
</bean>
```

2.2 毕业设计题目所属学院录入模块

（1）Hibernate 框架

首先建立数据库层，即 DAO。

Design.java 是数据库 DESIGN 表对应的持久化类，有三个属性，分别和 DESIGN 表中的三个字段对应：student 对应 DESIGN 表中的 STUDENT 字段，表示该毕业设计题目的负责人；topic 对应 DESIGN 表中的 TOPIC 字段，表示该毕业设计题目名称；college 对应 DESIGN 表中的 COLLEGE 字段，表示该毕业设计题目的所属学院。

```
//Design.java
@Entity
@Table(name = "DESIGN", schema = "Graduation")
public class Design {
    private String student;
    private String topic;
    private String college;
    //省略 getter 和 setter 方法
    //省略 equals 和 hashCode 重载方法
}
```

Table.hbm.xml 是 Hibernate 的实体类映射文件，在该文件中书写了数据库 Graduation 中的 DESIGN 表和 Design.java 实体类的映射关系：student 属性映射到 DESIGN 表的 STUDENT 字段，为主键；topic 属性映射到 DESIGN 表的 TOPIC 字段，college 属性映射到 DESIGN 表中的 COLLEGE 字段。

```xml
<!-- Table.hbm.xml -->
<hibernate-mapping>
    <class name="Table.Design" table="DESIGN" schema="Graduation">
        <id name="student" column="STUDENT"/>
        <property name="topic" column="TOPIC"/>
        <property name="college" column="COLLEGE"/>
    </class>
</hibernate-mapping>
```

hibernate.cfg.xml 是 Hibernate 的配置文件，在该文件中书写了数据库配置信息以及映射文件的注册信息，数据库的连接后期转交 Spring 框架处理，所以这里我已经把数据库配置信息转交到了 Spring 框架的配置文件中。

```xml
<!-- hibernate.cfg.xml -->
<hibernate-configuration>
  <session-factory>
    <property name="dialect">org.hibernate.dialect.MySQLDialect</property>
    <mapping class="Table.Design"/>
    <mapping resource="Table/Table.hbm.xml"/>
  </session-factory>
</hibernate-configuration>
```

AddCollegeToTopicDao.java 是毕业设计题目所属学院录入模块的 DAO 类，该类有一个 addCollege 方法，传入题目负责人和所属学院修改数据，然后用 sessionFatory 对象打开一个 session，并通过 Design.java 数据库表映射实体类和传入的题目负责人查询数据库 DESIGN 表中匹配负责人的数据。把查询到的结果和传入的需要修改的所属学院数据进行比对，如果相同，表示重复录入，返回 false；否则打开一个事务，利用传入的修改数据来保存或更新指定的毕业设计题目，最后提交事务并关闭 session，返回 true。

```java
//AddCollegeToTopicDao.java
//往已有毕业设计题目中录入学院名 Dao
public class AddCollegeToTopicDao {

    private Design design;  //一个 Design 对象
    private SessionFactory sessionFactory;  //session 工厂对象

    //用 sessionFactory 打开一个 session,然后根据账号传参查询数据库
    //如果查询到的题目学院名称和传参相同,表示重复录入,返回 false
     //否则开启一个事务并更新学院名,返回 true
     public boolean addCollege(String student, String college) {
         Session session = sessionFactory.openSession();
         design = session.get(Design.class, student);
         if(design.getCollege().equals(college)) {
             session.close();
             return false;
         } else {
             design.setCollege(college);
             Transaction ts = session.beginTransaction();
```

```
            session.saveOrUpdate(design);
            ts.commit();
            session.close();
            return true;
        }
    }
}
```

QueryTopicDao.java 是毕业设计题目所属学院录入模块的另一个 DAO 类，该类为进行毕业设计题目所属学院录入时提供题目筛选功能，它有两个方法：

一个是 QueryByKeyword 方法，传入筛选关键字，然后用 sessionFatory 对象打开一个 session，并通过建立 Hibernate 框架中的 HQL 语句，把数据库中毕业设计题目负责人、题目名称或所属学院名称中包含有关键字的所有匹配项查出来，然后关闭 session 对象，最后通过保存在 ArrayList 对象中返回。

另一个是 QueryAll 方法，该方法没有传参，表示不进行筛选，而是直接使用 HQL 语句查询数据库中的所有毕业设计题目，查询结束后关闭 session 对象，然后把查询到的结果保存在 ArrayList 对象中返回。

```
//QueryTopicDao.java
//按关键字查询毕业设计题目
public class QueryTopicDao {

    private ArrayList<Design> topics;   //查询毕业设计题目结果集
    private SessionFactory sessionFactory;  //session 工厂对象

    //用 sessionFactory 打开一个 session,然后根据账号传参查询数据库
    //根据关键字查询并返回数据库中负责人姓名、题目名称或所属学院名含有关键字的匹配项
     public ArrayList<Design> QueryByKeyword(String keyword) {
         Session session = sessionFactory.openSession();
         String HQL = "from Design where concat(student,topic,college) like :key
word";
         Query query = session.createQuery(HQL);
         query.setParameter("keyword","%" + keyword + "%");
         topics = (ArrayList<Design>) query.list();
         session.close();
         return topics;
     }

     //查询并返回数据库中全部毕业设计题目
     public ArrayList<Design> QueryAll() {
         Session session = sessionFactory.openSession();
         String HQL = "from Design ";
         Query query = session.createQuery(HQL);
         topics = (ArrayList<Design>) query.list();
         session.close();
         return topics;
     }
}
```

（2）Struts 框架

利用 Struts2 框架搭建基本的 MVC 模型，即前端用 JSP 显示，然后提交表单

到后端对应 Action 业务处理类，Action 类调用 DAO 层访问数据库并得到结果，然后处理结果，最后根据结果的不同跳转到不同的页面。Struts2 的监听器配置见登录模块中的 web.xml 配置文件所示。

addCollegeToTopic.jsp 是毕业设计题目所属学院录入模块的主要页面。

其最上方会查询 session 中的 user 对象的 student 属性，查询当前登录系统的使用者，并对使用者表示欢迎；

然后是一个毕业设计题目的筛选表单，用户可以通过关键字来筛选用户需要的毕业设计题目，这样可以大大地减少显示的信息，让用户更快地检索到需要录入所属学院的条目，然后进行毕业设计题目所属学院的录入操作；

再下来就是毕业设计题目的展示表格，会一条一条地显示数据库中的毕业设计题目信息。

最后就是两个操作按钮，左边的是返回系统菜单的按钮，可以返回到登录系统后的初始界面。右边的是退出登录按钮，可以让用户退出登录，即注销登录信息，系统会跳转至登录页面等待用户再次登录。

```
<!-- addCollegeToTopic.jsp -->
1. <h2>欢迎 ${sessionScope.user.student } 使用毕业设计题目系统!</h2><hr>
2. <h3>筛选毕业设计题目:</h3>
3. <form action="filterTopic.action" method="post">
4.     筛选关键字: <input type="text" name="keyword">
5.     <input type="submit" value="筛选">
6. </form><hr>
7. <table border="2" style="text-align: center">
8.     <!—省略表头代码 -->
9.     <%
10.         Map se = ActionContext.getContext().getSession();
11.         ArrayList<Design> topics = (ArrayList<Design>) se.get("topics");
12.         for(int i = 0;i < topics.size();i++) {
13.     %>
14.     <tr>
15.     <form action="addCollege.action" method="post">
16.         <td><%=i+1%></td>
17.         <td><input type="hidden" name="student" value=<%=topics.get(i).getStude
nt()%>><%=topics.get(i).getStudent()%></td>
18.         <td><%=topics.get(i).getTopic()%></td>
19.         <td><input type="text" name="college" value=<%=topics.get(i).getCollege
()%>></td>
20.         <td><input type="submit" value="录入"></td>
21.     </form>
22.     </tr>
23.     <%
24.         }
25.     %>
26.     <s:actionerror cssStyle="color:red"/>
27.     <s:actionmessage/>
28.     <form action="graduation.jsp" method="post">
29.     <input type="submit" value="返回菜单" style="font-size: 20px">  </form>
30.     <form action="quit.action" method="post">
31.     <input type="submit" value="退出" style="font-size: 20px">  </form>
32. </table>
```

进入毕业设计题目系统的题目所属学院录入模块，首先就是欢迎使用者的欢迎语，欢迎语中依然会显示使用者的用户名，直接对正在使用系统的用户进行欢迎。

然后是一个筛选毕业设计题目的筛选框，用户可以在其中键入关键字来筛选出系统中负责人姓名、毕业设计题目名称或者毕业设计题目所属学院中任意包含关键字的全部题目信息。

再来就是展示数据库中的毕业设计题目，如果用户使用了筛选功能，则只显示用户需要的那部分毕业设计题目。每道题目对应地后面的所属学院是可以编辑修改的，旁边还有对应的录入功能键。用户在编辑好需要更改的所属学院名称后，点击录入按钮即可把新数据写入到对应的题目中。具体样式如图 15 所示。

首次进入毕业设计题目所属学院录入模块，会默认显示数据库中的全部毕业设计题目。该系统使用 Hibernate 框架，对数据库的表的查询具有可持久化的性质，即查询数据非常快速。

首次进入毕业设计题目所属学院录入模块页面，如图 15 所示。

图 15. 首次进入毕业设计题目所属学院录入模块

struts.xml 是 Struts2 的配置文件，在该配置文件中注册毕业设计题目所属学院录入模块中需要用到的两个 Action 类以及对他们进行一些跳转逻辑地址的映射。由于后期整合 Struts2 框架和 Spring 框架后，Action 类的生成转交 Spring 框架进行，所以这里的 class 为别名，与 Spring 框架中的相应配置信息对应。

```
<!-- sturts.xml -->
<struts>
<package name="struts2" extends="struts-default">
    <action name="addCollege" class="addCollege">
        <result name="success">/WEB-INF/JSP/addCollegeToTopic.jsp</result>
        <result name="error">/WEB-INF/JSP/addCollegeToTopic.jsp</result>
    </action>
    <action name="filterTopic" class="filterTopic">
        <result name="success">/WEB-INF/JSP/addCollegeToTopic.jsp</result>
     </action>
</package>
</struts>
```

AddCollegeToTopicAction.java 是毕业设计题目所属学院录入模块中的处理类，在该类中有负责人姓名、学院名、添加学院名的 Dao 对象、查询题目的 Dao 对象、以及毕业设计题目查询结果集对象。其中负责人姓名和学院名属性对应前端 JSP 传来的表单数据；添加学院名的 Dao 对象用于为指定的毕业设计题目添加所属学院信息；查询题目的 Dao 用于添加学院名称后对前端显示数据进行更新操作；毕业设计题目查询结果集用于保存查询题目 Dao 的返回结果，并通过保存在 session 中最终更新到前端。

在 execute 方法中，首先调用 addCollegeToTopicDao 对象的 addCollege 方法，传入前端表单传来的题目负责人和所属学院修改信息，得到一个 boolean 型 addCollege。

然后调用 queryTopicDao 对象的 QueryAll 方法，查询添加了学院名称后的数据库中的毕业设计题目，得到的结果保存在毕业设计题目查询结果集中。开启 session 对象，把查询结果保存。

addCollege 如果为真，表示录入成功，跳转至成功页面；如果为假，则返回"录入失败!重复录入或录入信息不合法!"的红色的错误信息到前端，等待用户重新录入题目的所属学院名称。

```
//AddCollegeToTopicAction.java
//往已有毕业设计题目中录入学院名
public class AddCollegeToTopicAction extends ActionSupport {

    private String student;              //负责人姓名
    private String college;              //学院名
    private AddCollegeToTopicDao addCollegeToTopicDao;//添加学院 Dao
    private QueryTopicDao queryTopicDao;//查询 Dao
    private ArrayList<Design> topics;    //毕业设计题目查询结果集

     @Override
     public String execute() {
         boolean addCollege = addCollegeToTopicDao.addCollege(student,college);
         //添加学院后更新页面的毕业设计题目展示表
         topics = queryTopicDao.QueryAll();
         Map session = ActionContext.getContext().getSession();
         session.put("topics",topics);
         //addCollege 为真,录入成功,数据已更新到数据库
         //addCollege 为假,录入失败,在页面显示录入失败等红字提示
```

```
20.         if(addCollege) {
21.             addActionMessage("录入成功!");
22.             return "success";
23.         } else {
24.             addActionError("录入失败!重复录入或录入信息不合法!");
25.             return "error";
26.         }
27.     }
28. }
```

进行毕业设计题目所属学院的信息录入，选择在录入系统时没有添加所属学院的题目，即负责人是气温的“基于天使轮投资的风险评估”毕业设计，在其后的所属学院中写上“金融学院”，然后点击其后方对应的录入按钮。详细情况如图 16 所示。

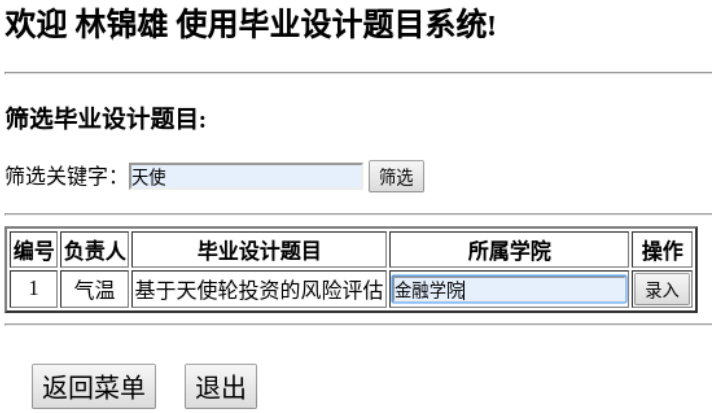

图 16. 进行题目所属学院信息录入

在毕业设计题目所属学院录入功能中对题目的所属学院进行录入，录入成功后，会反馈录入成功的提示语，并实时更新题目展示表中的数据内容。如图 17 所示。

图 17. 录入成功

在毕业设计题目所属学院录入功能中对题目的所属学院进行录入时，如果录入的信息与数据库中保存的信息一致，则系统会认为用户在重复录入；另外，如果录入的信息不合法，系统也会检测出来，然后统一反馈“录入失败!重复录入或录入信息不合法”的红色提示语句，提示用户操作失败，请修正更改信息后重新进行操作。如图 18 所示。

图 18. 重复录入

FilterTopicAction.java 是毕业设计题目所属学院录入模块中的过滤类，用于过滤前端显示页面中过多的毕业设计题目，以达到快速检索需要录入所属学院的题目的目的。该类有过滤关键字、查询题目 Dao 以及毕业设计题目过滤结果集三个属性。其中过滤关键字对应前端检索框中的关键字；查询题目 Dao 用于查询题目负责人、题目名称或题目所属学院名称中含有关键字的题目；查询结果保存在结果集中。

execute 方法中首先判断过滤关键字是否为空，如果为空则调用 queryTopicDao 对象中的 QueryAll 方法，查询整个数据库并返回全部毕业设计题目信息。否则调用 QueryByKeyword 方法，传入关键字，把符合条件的题目结果返回。最后打开 session，把查询结果集更新到 session 中，最后跳转到原页面。

```
1. //FilterTopicAction.java
2. //过滤指定毕业设计题目供操作
3. public class FilterTopicAction extends ActionSupport {
4.
5.     private String keyword;              //过滤关键字
6.     private QueryTopicDao queryTopicDao;//查询 Dao
7.     private ArrayList<Design> topics;   //毕业设计题目过滤结果集
8.
9.     @Override
10.     public String execute() {
11.         //如果过滤关键字为空,表示查询数据库中全部毕业设计题目
12.         //根据关键字查询并返回数据库中负责人、题目名称或所属学院名含有关键字的匹配项
13.         if(keyword == null || keyword.equals("")) {
14.             topics = queryTopicDao.QueryAll();
15.         } else {
16.             topics = queryTopicDao.QueryByKeyword(keyword);
17.         }
18.         Map session = ActionContext.getContext().getSession();
19.         session.put("topics",topics);   //把过滤结果更新到 session
20.         return "success";
21.     }
22. }
```

在上面的录入功能测试中可以看到在录入之前，筛选关键字有“天使”的关键词存在，并且题目展示表只有一项，其中题目名称中包含了“天使”这个词，这就是使用了该模块中的筛选毕业设计题目的功能。

在筛选关键字输入框中输入关键字或关键词，点击筛选按钮，系统就会自动查询数据库中负责人姓名、毕业设计题目名称、题目所属学院中任意包含有关键字或关键词的所有项目，并把筛选的结果更新到题目展示表中。筛选成功的结果，如图 19 所示。

图 19. 通过关键字筛选题目

（3）Spring 框架

把 Hibernate 框架和 Spring 框架进行整合，把 Struts2 框架和 Spring 框架进行整合，由 Spring 框架控制 Hibernate 框架中对数据库的连接和映射，Struts2 框架中的各种对象的生成也转交 Spring 处理，例如 Struts2 框架中的 Action 需要创建 Hibernate 框架中的数据库访问类即 DAO 层，这里将 DAO 层的对象创建转交给 Spring 框架完成，还有 Struts2 框架中各种类的装配也转交 Spring 进行处理，可以大大地降低程序的耦合度。Spring 的监听器配置见登录模块中的 web.xml 配置文件所示。

applicationContext.xml 是 Spring 框架的配置文件，用于整合 Hibernate 框架和 Struts2 框架，让 Hibernate 框架需要做的数据库配置信息和连接操作，实体类与数据库表的映射配置，实体类的生成都交由 Spring 框架装配；再整合 Struts2 框架，让 Struts2 框架中需要生成的各种 Action 以及需要创建的各种 Hibernate 框架中的 DAO 层对象都转交 Spring 框架装配。

首先是装配一个 Table.Design 实体类的 design 对象。然后装配 Struts2 的录入学院模块的 Action 和 DAO 以及过滤题目模块的 Action 和 DAO，并为他们注入 sessionFactory 对象，为 Action 注入 DAO 层的对象。还有数据库连接的配置信息装配注入到 sessionFactory 中。这样就完成了 Spring 框架对 Hibernate 框架以及 Struts2 框架的整合。

```
<!-- applicationContext.xml -->
<!-- Bean -->
<bean name="design" class="Table.Design" scope="prototype"></bean>

<!-- 录入学院模块 -->
<bean name="addCollegeToTopicDao" class="DAO.AddCollegeToTopicDao" scope="prototype">
    <property name="sessionFactory" ref="sessionFactory"></property>
</bean>
<bean name="addCollege" class="Action.AddCollegeToTopicAction" scope="prototype">
    <property name="addCollegeToTopicDao" ref="addCollegeToTopicDao"></property>
```

```
    <property name="queryTopicDao" ref="queryTopicDao"></property>
</bean>

<!-- 过滤题目模块 -->
<bean name="filterTopic" class="Action.FilterTopicAction" scope="prototype">
    <property name="queryTopicDao" ref="queryTopicDao"></property>
</bean>

<!-- 退出模块 -->
<bean name="quit" class="Action.QuitAction" scope="prototype"></bean>

<!-- sessionFactory -->
<bean name="sessionFactory" class="org.springframework.orm.hibernate5.LocalSessionFactoryBean" scope="prototype">
    <property name="dataSource" ref="dataSource"></property>
    <property name="mappingLocations" value="classpath*:Table/Table.hbm.xml"></property>
    <property name="mappingResources" value="hibernate.cfg.xml"></property>
</bean>
<bean name="dataSource" class="org.apache.commons.dbcp2.BasicDataSource" scope="prototype">
    <property name="driverClassName" value="com.mysql.jdbc.Driver"></property>
    <property name="url" value="jdbc:mysql://localhost:3306/Graduation?characterEncoding=UTF-8"></property>
    <property name="username" value="root"></property>
    <property name="password" value="bingjie123"></property>
</bean>
```

2.3 退出模块

该模块主要用于用户在使用系统结束后的退出登录操作。QuitAction.java 的 execute 方法会获得 session，然后删除 session 中保存的 key 为 user 的值，然后新系统中的其他页面在监测是否登录时就会监测到未登录。success 字符串对应的逻辑地址为 login.jsp，即登录界面，退出登录后都会跳转至登录页面等待用户再次登录。

```
//QuitAction.java
public class QuitAction {
    public String execute() {
        Map session = ActionContext.getContext().getSession();
        session.remove("user"); //删除 session 中的 key 为 user 的数据
        return "success";
    }
}
```

2.4 登录过滤模块

该模块用于过滤未登录就访问系统的非法操作，即当用户未登录就想使用 URL 直接访问系统，该过滤器就会监测到用户未登录，然后强制跳转至登录页面请用户登录后再访问系统。

LoginFilter.java 类使用传参 ServletRequest 类型的对象获得 session 对象，然后检测 key 为 user 是否存在，如果不存在或者值为空则过滤，跳转至登录页面。如果 user 存在，就不进行过滤操作。

LoginFilter.java 类通过注解方式注册需要过滤的 jsp 和 action 类。

```
1. //LoginFilter.java
2. //登陆检查过滤器
3. @WebFilter(urlPatterns = {"/graduation.jsp","/addCollegeToTopic.jsp","/entryTopic
.jsp","/queryTopic.jsp","/showTopic.jsp","/query.action","/entry.action","/addColle
ge.action"})
4. public class LoginFilter implements javax.servlet.Filter {
5.
6.     public void doFilter(javax.servlet.ServletRequest req, javax.servlet.ServletR
esponse resp, javax.servlet.FilterChain chain) throws javax.servlet.ServletExceptio
n, IOException {
7.         HttpServletRequest request = (HttpServletRequest) req;
8.         HttpSession session = request.getSession();
9.         User user = (User) session.getAttribute("user");
10.          //查询 session 中是否存在 user 对象
11.          //对象不存在或为用户名为空,表示未登陆,跳转至登陆界面
12.          if(user == null || user.getStudent().equals("")) {
13.              request.getRequestDispatcher("/login.jsp").forward(req, resp);
14.          } else {
15.              chain.doFilter(req, resp);
16.          }
17.     }
18. }
```

3、结论

系统采用当前比较成熟的 web 开发框架 SSH 和 Tomcat 技术实现设计与开发，主要用到了比较实用的 MVC 模型，使得各模块具有相当的独立性，因此系统的可重用性也比较高。用 Java 语言作为开发语言，也使程序具有很强的可移植性。

毕业设计题目查询系统是网页应用系统，可以部署于服务器中，实现了对毕业设计题目信息的科学、高效的管理，此系统使得使用者可以很方便地添加毕业设计题目并对其进行管理。相比较过去利用纸质和人力管理模式，开发利用此系统不仅节省了人力物力，还简化了毕业设计题目档案的查询流程。

该系统具有完整的账号登录和注册功能，使用登录和注册功能时，使用者的各种操作系统都会有相应的信息反馈，以帮助使用者更好地使用功能。

该系统还实现了针对毕业设计题目所属学院的录入、毕业设计题目的录入和毕业设计题目信息查询等功能。毕业设计题目可以很方便地录入系统；所有已录入系统地题目都可以在后期对所属学院等信息进行修改；该系统还有相当完善方便的查询功能，只要用户提供关键字或关键词，系统即可以查找到题目负责人姓名、题目名称、所属学院名称中包含有关键字或关键词的全部项目。

在此次开发过程中，遇到不少难点，例如模糊查询的设计方式，如何才能更

好地查询到符合用户需求的结果，如何提高查询速度等等，都考虑了相当长的时间，并设计出了几种方案来作比较。最后也是一步一步地修改、调试，找到最合适地方案来操作数据库。我知道，短时间开发的系统肯定是不完善的，我需要在测试和使用中去发现问题，并不断地完善这个系统，在这一点上，很多同学也给了我不少的设计意见，如表单的设计风格，前端错误的反馈样式等等，我都一一比较并接受觉得合适的意见。

在设计题目所属学院录入模块的筛选题目功能时，我是直接在后端处理数据并刷新整个页面。后期考虑到，当系统中的毕业设计题目数据量越来越大，那么查询整个数据库所需要的时间就不是 Hibernate 框架的可持久化性质能解决的了，可能会致使使用者点击筛选按钮后页面长期变为空白，直到后台查询数据库操作完成为止，这样对用户的体验极其不好。于是我便计划筛选功能使用异步提交表单，这样，用户在点击筛选按钮后仍然可以继续浏览页面的其他信息，等待后台查询完成后，查询结果再刷新到对应的区域，这样就可以在不整体刷新页面，保证用户的使用体验。

在实现过程中，虽然遇到了不少困难，但我通过自己查资料、以及在指导老师陆洁茹老师的指导下以及和同学相互交流中，设计出解决方案并成功实现本系统。

经过这次的软件开发，确实学到了不少的东西，MVC 开发模式、JAVA SSH 框架、JSP 前端开发、JDBC 的网页应用等等，同时也深感自己知识的欠缺。在以后的学习中，一定要继续坚持不段地学习专业知识及相关的非专业知识，只有这样才能紧跟时代的潮流。

参考文献

[1] 牛德雄．基于 MVC 的 JSP 软件开发案例教程[M]．北京：清华大学出版社，2014．

[2] 郭克华．JavaEE 程序设计与应用开发（第 2 版）[M]．北京：清华大学出版社，2017．

[3] Bruce Eckel．Java 编程思想（第 4 版）[M]．北京：机械工业出版社，2007．

[4] Craig Walls．Spring 实战（第 4 版）[M]．北京：人民邮电出版社，2016．

[5] 褚久良．Web 前端开发技术 （第 2 版）[M]．北京：清华大学出版社，2013．

[6] 萨师煊．数据库系统概论（第 5 版）[M]．北京：高等教育出版社，2014．

[7] 谌卫军．Java 程序设计（第 1 版）[M]．北京：清华大学出版社，2016．