关注微信公众号

👇👇👇

CoderGeshu

目录

一、前言.....2
二、系统设计概述.....2
1. 系统设计愿景.....2
2. 系统功能描述及要求.....2
3. 系统数据库要求.....3
4. 系统客户端程序设计要求.....4
三、需求分析.....4
1. 系统参与者.....4
2. 系统用例图.....4
3. 主要用例规约.....7
四、系统设计.....10
4.1 软件平台设计.....10
4.2 系统架构设计.....10
4.3 系统模块结构设计.....11
4.4 数据库设计.....13
4.4.1 概念结构设计.....13
4.4.2 逻辑结构设计.....14
4.4.3 物理结构设计.....16
4.5 系统详细设计与界面设计.....16
4.5.1 登录系统详细设计.....16
4.5.2 用户主界面详细设计.....18
4.5.3 用户详细信息显示页面设计.....19
4.5.4 显示用户信息表格的页面设计.....19
4.5.5 修改密码功能设计.....20
4.5.6 学生填报志愿和查看志愿功能设计.....20
4.5.7 导师处理学生志愿功能设计.....21
4.5.8 系管理员主要功能设计.....21
4.5.5 研究生院管理员主要功能设计.....22
五、系统核心模块描述.....24
5.1 使用 PreparedStatement 执行可变参数的更新语句.....24
5.2 登录验证.....26
5.3 切换用户功能.....27
六、总结.....28
6.1 系统已经实现的功能及可扩充的方向.....28
6.1.1 系统实现的用户功能.....28
6.1.2 可以扩充的方向.....29
6.2 遇到的问题及其解决方法.....30
6.3 开发过程中学到的技术.....32

一、前言

随着信息技术的迅速发展，多媒体计算机、网络为代表的信息技术给教育的发展带来了新的生机，为传统的教学方式注入了活力，给其带来了重大的革命。

为顺应网络的普及以及教学数字化的高速发展，传统的研究生导师选择方式早已不能满足教师和学生的信息需求，研究生导师双选系统适应当代大学教学需要，迎合信息时代的发展潮流，立足于方便教师同学进行信息化管理、信息获取、资源共享以及及时进行互动交流，极大的弥补了传统方式对于资源的获取以及使用者之间交流上的不足。

研究生导师双选系统可以帮助减轻老师、学生们的工作量，对导师学生的信息通过互联网的管理更方便更科学更直观可，方便校方教务的管理，也方便了学生。

二、系统设计概述

1. 系统设计愿景

研究生导师双选系统（以下统称为“本系统”）是常见的计算机信息管理系统。它的主要任务是实现研究生和导师的关系选择，对学生、导师信息进行日常的管理，在线上能方便的实现导师和研究生各自具有的功能，能够迅速准确地完成各种计算和汇总，快速打印出报表。充分发挥管理系统的高效、完全、可靠、便捷的性能，减少教务人员、学生以及导师的工作量。

2. 系统功能描述及要求

每个用户利用自己的角色及相应的账号密码登录系统。研究生需要选择自己的导师，导师也需要选择学生；管理员需要管理学生、导师的信息数据，并最终确定学生导师关系。每个研究生选导师有三个志愿，而且导师所带学生上限由管理员设定。

本系统的用户及其功能详细要求：

（1）研究生院管理员（超级管理员）

1） 登录系统；

2） 学生基本信息批量导入（单个录入），维护学生信息；

3） 导师基本信息批量导入（单个录入），维护导师信息；

4） 管理（增加，维护）各学院级管理员信息；
5） 查看学生导师关系最终选择情况；
6） 导出学生导师选择志愿表。
7） 维护个人信息；

（2）学院级管理员（权限只限于本学院）

1） 登录系统；
2） 维护学生基本信息（不具备信息导入功能）；
3） 维护导师基本信息（不具备信息导入功能）；
4） 设定导师所带学生人数的上限；
5） 查看各专业导师、学生信息；
6） 查看学生导师选择志愿信息，最终确定学生导师选择关系；
7） 导出学生导师选择志愿表。
8） 维护个人信息；

（3）学生：

1） 登录系统；
2） 查看本专业导师信息（包含导师基本信息，导师现在所带学生人数，导师所带人数剩余名额）；
3） 填写导师志愿表（三个志愿必填），不可选择所带人数已满的导师，当提交志愿表后不可修改；
4） 查看导师志愿表的审核状态（审核中、审核通过、审核未通过）；
5） 导师志愿表审核通过后可以查看最终确定的导师信息；
6） 导师志愿表审核不通过时系统允许重新填写导师志愿表；
7） 维护个人信息（账号不可修改）；

注：导师所带人数是学院级管理员最终确定关系后的人数，并不等同于选择此导师的学生人数。

（4）导师：

1） 登录系统；
2） 查看选择自己的学生信息；
3） 选择所带学生（在自己的数量上限范围内），等待院级管理员的确认；
4） 查看所带学生的信息；
5） 维护个人信息（账号不可修改）。

3. 系统数据库要求

（1）为了保证数据库系统的正确性、完备性和一致性，就必须进行数据完整性

设计。就本设计而言应考虑实施如下数据完整性：

1） 给表设置主键和外键约束，如用户账号。

2） 设定缺省约束。如教师、学生的性别。

3） 设置非空约束。如教师、学生的姓名。

4） 其他数据库完整性设计。

（2）MySQL数据库对象设计

为充分发挥数据库的效能，保证数据库的安全性，提高数据库管理系统的执行效率，可以考虑使用视图、存储过程及表的触发器来实现某些功能。本设计可考虑如下数据库对象。

1） 为提高检索性能，为表创建索引。

2） 用户信息视图。设计一个视图，为不同权限的人显示不同的用户信息。

3） 其他数据库对象设计。

4. 系统客户端程序设计要求

使用 Java EE 程序设计以及 MySQL 数据库等编程技术实现本系统（B/S 架构），要求界面简洁友好、操作简单。

三、需求分析

1. 系统参与者

用系统功能描述及要求可知，本系统的参与者有：研究生院管理员（超级管理员）、院级管理员、学生、导师。

2. 系统用例图

因为此系统由不同参与者使用，为了表述更加清晰与准确，将系统用例图按照不同用户划分成子用例图，并严格依据由系统功能描述及要求绘制。

（1）系统用户用例图

系统中所有用户都具有登录和维护个人信息功能，所以抽象出一个用户参与者实现这两个功能，其他具体用户都继承自此抽象用户，也具备登录和维护个人信息功能。如图 3-2-1 所示。

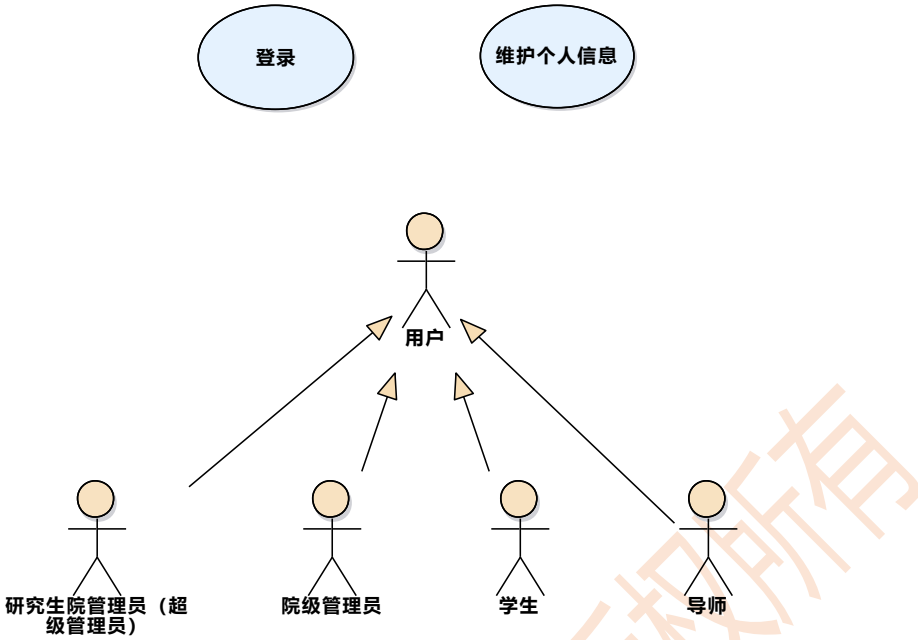

图 3-2-1 系统用户用例图

（2）研究生院管理员（超级管理员）用例图

根据对研究生院管理员功能的描述与要求，设计研究生院管理员用例图，如图 3-2-2 所示。

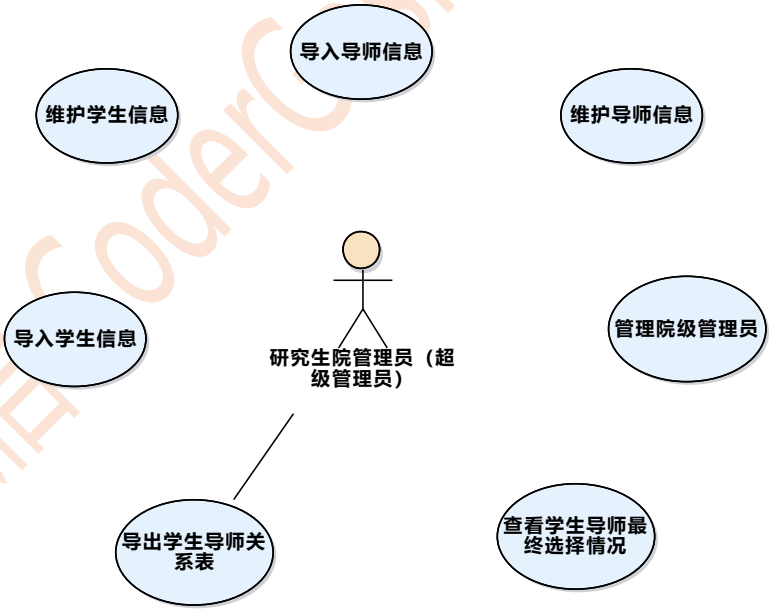

图 3-2-2 研究生院管理员（超级管理员）用例图

（3）院级管理员用例图

根据对院级管理员功能的描述与要求，设计院级管理员用例图，如图 3-2-3 所示。

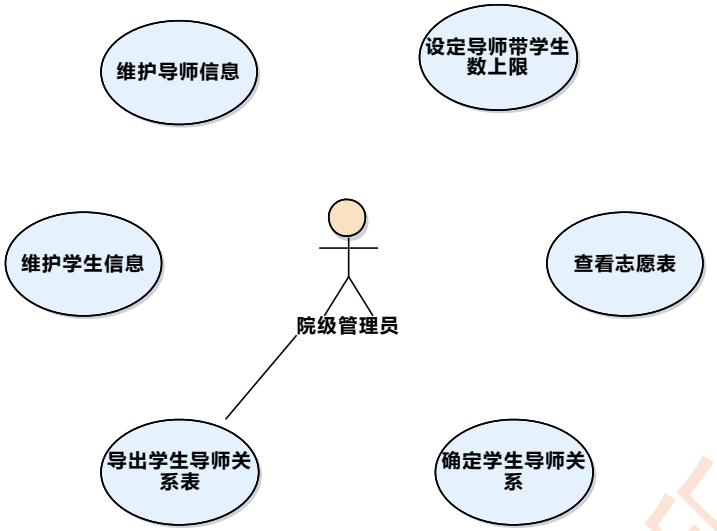

图 3-2-3 院级管理员用例图

需要指出的是，各院级管理员只允许管理本学院的导师和学生信息。

（4）学生用例图

根据对学生功能的描述与要求，设计学生功能用例图，如图 3-2-4 所示。

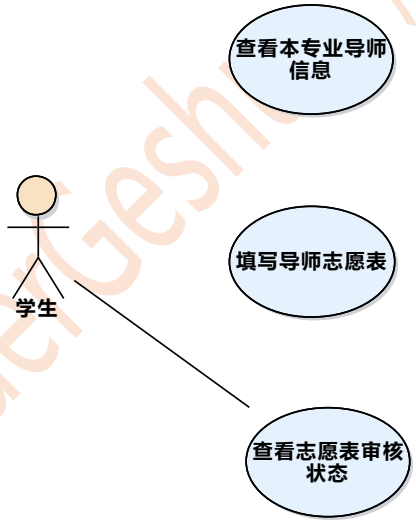

图 3-2-4 学生用例图

（5）导师用例图

根据对导师功能的描述与要求，设计导师功能用例图，如图 3-2-5 所示。

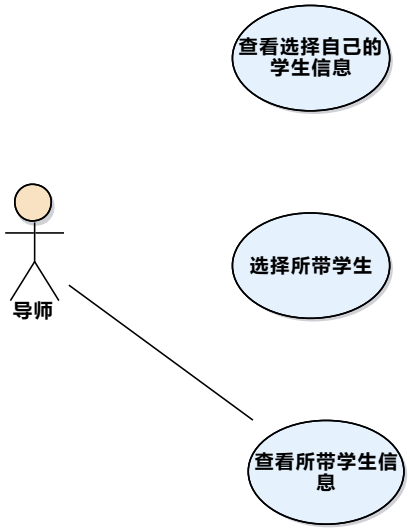

图 3-2-5 导师用例图

3. 主要用例规约

通用用例规约：

（1）登录

用例说明：登录用例允许合法用户登录系统，同时保证不同用户角色（“超管”、“系负责人”、“学生”、“导师”）使用系统登录用例后进入不同的功能权限界面端。

1） 基本事件流：用户按键输入账号和密码，并选择用户角色（学生、导师、系负责人、超管）。用户按“确认登录”按钮进行登录。系统确认是否有效。如果有效，系统承认这次登录，登录用例结束。

2） 可选事件流：用户可以在按“确认登录”按钮之前的任何时刻可以清除账号、密码或重新选择用户角色。

3） 特殊需求：应设计合理、简洁的页面实现要求的功能，更高一层的要求是页面美观。

4） 前置条件：用户进入登录页面。

5） 后置条件：用例执行成功，提示用户登录成功的信息。用户进入相应角色的功能界面。

（2）维护个人信息

1） 基本事件流：展示用户的个人资料信息，并允许用户修改可修改的信息（账号不可修改），用户点击确定修改按钮后更新用户的个人资料。如果信息更新成功，则展示更新后的信息，用例结束。

2） 可选事件流：在用户未提交修改信息之前用户可以取消修改资料，用户提交更新信息之前可以返回主功能页面，当前信息不做任何修改。

3） 异常事件流：如果用户修改了密码，则需要两次验证新密码，如果两次密码不一致，将提醒用户并且不对用户信息做出任何修改。

4） 前置条件：用户选择维护个人信息。

5） 后置条件：如果用户更新信息满足，则更新数据库，显示用户新信息。

超管（研究生院管理员）的用例：

（1）导入学生信息、导入导师信息

1) 基本事件流：超管需使用.xls 文件格式并以某一确定的内容结构导入学生或导师信息文件，文件导入成功，则更新数据库学生信息。

2) 异常事件流：如果上传文件不符合基本事件流，则给出提示信息，并放弃导入。

3) 前置条件：超管选择导入学生/导师信息。

4) 后置条件：更新数据库，刷新学生和导师信息页面。

（2）管理院级管理员

1) 基本事件流：超管可以查看所有系负责人（院级管理员）的信息，可以增加修改或删除某一学院的系负责人，取消其权限。

2) 前置条件：超管进入院级管理员界面。

3) 后置条件：对超管所做的操作更新数据库信息，并刷新页面。

系负责人用例：

（1）设定导师可带学生人数上限

1) 基本事件流：默认导师可带人数为 3 位，院级管理员可对其院下的导师进行能力重分配，即可以重新定义导师所带学生人数。系负责人首先选择查看导师信息，然后可以在所带人数一栏中进行修改，修改提交之后，更新数据库并刷新页面。

2) 异常事件流：导师能力上限需要设定，不能超过上限人数，否则放弃更改。

3) 前置条件：系负责人选择修改导师所带人数。

4) 后置条件：更新数据库，刷新页面信息。

（2）查看学生导师双向选择志愿表

基本事件流：系负责人可以查看学生导师目前选择的情况，显示学生或导师的志愿选择，并查看相关结果。

（3）确定学生导师选择

1) 基本事件流：默认在导师接收学生的志愿之后就设定当前学生和导师的关系，但是有时会有所偏差，为了更好的更合理的分配，系负责人可以做出微调，重新分配或确定学生与导师的关系。

2) 异常事件流：如果系负责人为一位所带学生数已经达到上限的导师分配

学生，则产生错误提示，不予分配。

3) 前置条件：系负责人选择做出微调。

4) 后置条件：对做出的修改成功的信息进行入库更新，并刷新页面。

超管和系负责人共同的用例：

（1）维护学生信息

基本事件流：可以查看学生的基本信息，查看是否选择了导师，可以查看其导师信息。可以对学生基本信息做出修改。

（2）维护导师信息

基本事件流：可以查看导师信息，查看所带学生信息。可以对导师信息做出修改。

（3）查看学生导师选择情况

在此页面可以查看导师和学生选择志愿的情况

（4）导出学生导师关系表

基本事件流：选择导出学生导师关系表，则按照一定的格式导出 xls 文件，保存到本地文件。

学生用例：

（1）查看本专业导师信息

基本事件流：在专业导师界面，学生可以查看到所在系的导师列表，以便于学生找到自己心仪的导师。点击教师的详细信息可以查看更详细的教师信息内容。专业导师信息中包括可以带的人数限制以及目前已经带的人数。

（2）填写导师志愿表

1) 基本事件流：学生最多可以填报三个志愿，填报志愿时志愿不可以为空，不同学生的志愿信息可以相同。志愿填报成功之后，在志愿填报时间段内，学生还可以对已填报的志愿信息进行多次修改。学生可以浏览志愿的审核状态，以便及时修正。

2) 异常事件流：如果某学生选择了已经带满学生的导师，则系统给出提示不予其选择。

导师用例：

（1）查看选择自己的学生信息

基本事件流：导师可以查看“选择我的”的学生列表，并可以查看其详细信息，导师可以执行拒绝或者接收来选择是否确定此志愿关系，如果接收，则导师所带人数中就增加此同学，否则不变。

四、系统设计

4.1 软件平台设计

研究生导师双选系统的开发和运行环境如下：

(1) 操作系统：研究生导师双选系统可以运行在 windows 系列桌面操作系统之上。

(2) 支撑软件：本系统选用 IntelliJ IDEA 2020.03 版本作为开发工具，DBMS 选用 MySQL 5.5.62 版本。

(3) CASE 平台：教学辅助系统的分析、设计、实现和部署模型是在 Enterprise Architect 14 建模环境下创建的。数据库设计采用了 Process On 网页软件。

4.2 系统架构设计

应用架构模式：MVC 模式

（1）MVC 架构模式的内容

MVC 指 MVC 模式的某种框架，它强制性的使应用程序的输入、处理和输出分开。使用 MVC 应用程序被分成三个核心部件：模型、视图、控制器。它们各自处理自己的任务。

1）模型

模型表示企业数据和业务规则。在 MVC 的三个部件中，模型拥有最多的处理任务。被模型返回的数据是中立的，就是说模型与数据格式无关，这样一个模型能为多个视图提供数据，由于应用于模型的代码只需写一次就可以被多个视图重用，所以减少了代码的重复性。

2）视图

视图是用户看到并与之交互的界面。MVC 好处是它能为应用程序处理很多不同的视图。在视图中其实没有真正的处理发生，不管这些数据是联机存储的还是一个雇员列表，作为视图来讲，它只是作为一种输出数据并允许用户操纵的方式。

3）控制器

控制器接受用户的输入并调用模型和视图去完成用户的需求，控制器本身不输出任何东西和做任何处理。它只是接收请求并决定调用哪个模型构件去处理请求，然后再确定用哪个视图来显示返回的数据。

（2） MVC 架构模式的优点

1）耦合性低

视图层和业务层分离，这样就允许更改视图层代码而不用重新编译模型和控

制器代码，同样，一个应用的业务流程或者业务规则的改变只需要改动 MVC 的模型层即可。因为模型与控制器和视图相分离，所以很容易改变应用程序的数据层和业务规则。

模型是自包含的，并且与控制器和视图相分离，所以很容易改变应用程序的数据层和业务规则。如果把数据库从 MySQL 移植到 Oracle，只需改变模型即可。一旦正确的实现了模型，不管数据来自什么数据库，视图将会正确的显示它们。由于运用 MVC 的应用程序的三个部件是相互独立，改变其中一个不会影响其它两个，所以依据这种设计思想能构造良好的松耦合的构件。

2）重用性高

随着技术的不断进步，需要用越来越多的方式来访问应用程序。MVC 模式允许使用各种不同样式的视图来访问同一个服务器端的代码，因为多个视图能共享一个模型，由于模型返回的数据没有进行格式化，所以同样的构件能被不同的界面使用。由于已经将数据和业务规则从表示层分开，所以可以最大化的重用代码了。模型也有状态管理和数据持久性处理的功能，例如，基于会话的购物车和电子商务过程也能被 Flash 网站或者无线联网的应用程序所重用。

3）生命周期成本低

MVC 使开发和维护用户接口的技术含量降低。

4）部署快

使用 MVC 模式使开发时间得到相当大的缩减，它使程序员（Java 开发人员）集中精力于业务逻辑，界面程序员（HTML 和 JSP 开发人员）集中精力于表现形式上。

5）可维护性高

分离视图层和业务逻辑层也使得 Web 应用更易于维护和修改。

6）有利软件工程化管理

由于不同的层各司其职，每一层不同的应用具有某些相同的特征，有利于通过工程化、工具化管理程序代码。控制器也提供了一个好处，就是可以使用控制器来联接不同的模型和视图去完成用户的需求，这样控制器可以为构造应用程序提供强有力的手段。给定一些可重用的模型和视图，控制器可以根据用户的需求选择模型进行处理，然后选择视图将处理结果显示给用户。

4.3 系统模块结构设计

（1）使用 JSP 编程技术实现视图层；

本系统使用 JSP 实现视图一层，要求对视图一层要做到前后端的分离，前端页面使用标签库等；因为视图层是直接呈现给用户的，所以对前端视图的要求是

尽可能的简洁明了，同时也应具有一定的美化。

（2） 基于 Java Servlet 实现控制器；

使用 Servlet 实现控制一层，它作为视图层和模型的纽带，可谓是至关重要的一环。它主要负责从前端页面接收请求，同时处理请求的逻辑，从模型层中提取整合相应的信息，最后以相应的形式返回给前端页面，实现用户和系统的良好交互。

（3） 基于 Java Bean 实现设计模型；

使用 Java Bean 实现模型一层，它主要发挥保存模型信息、提供模型信息的作用，应设置良好的接口供控制层来调用获取。

（4） 基于 JDBC 存取数据库；

使用 JDBC 连接数据库， 要求做到数据库分层设计，独立出连接数据库模块和信息实体模块以及对实体信息操作模块。在对数据库信息操作过程中，应设计合理的查询等语句防止 SQL 注入的发生，可以使用 PreparedStatement 来实现。

数据库连接模块调用层次图如图 4-3-1 所示。

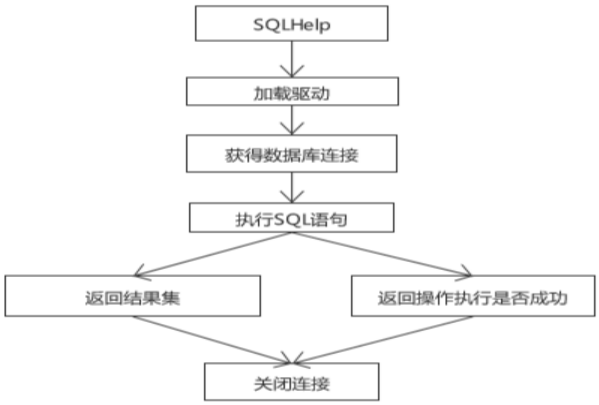

图 4-3-1 JDBC 连接数据库

基于 MVC 模式设计开发，整个项目的模块结构包图如图 4-3-2 所示。

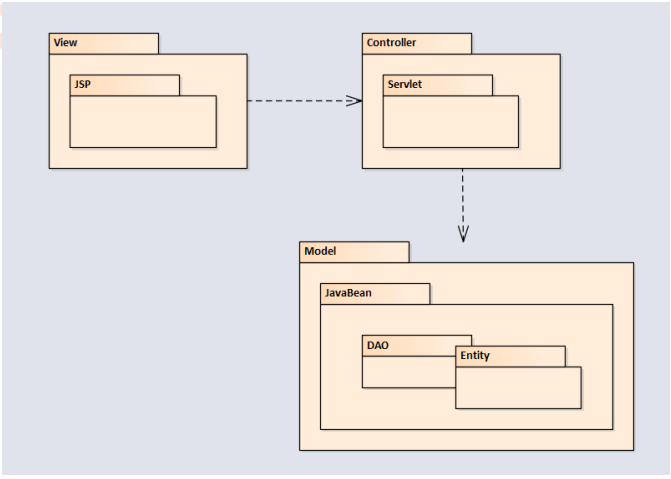

图 4-3-2 项目包结构

4.4 数据库设计

4.4.1 概念结构设计

（1）语义约束

1）设立多个学院，一个学院设立一位院级管理员,研究生院管理员可以管理所有学院；

2）一个学院设立多个专业，一个专业只能属于一个学院；

3）一个专业拥有多名学生，一名学生只能属于一个专业；

4）一个专业拥有多名导师，一名导师只能属于一个专业；

5）一名导师可带多名学生，一名学生只能最终确定一个导师；

6）一名学生可填三个志愿，老师选择接收或拒绝。

（2）系统 E-R 图

系统全部实体以及每个实体的属性，用 E-R 图表示，见图 4-4-1，图 4-4-2。

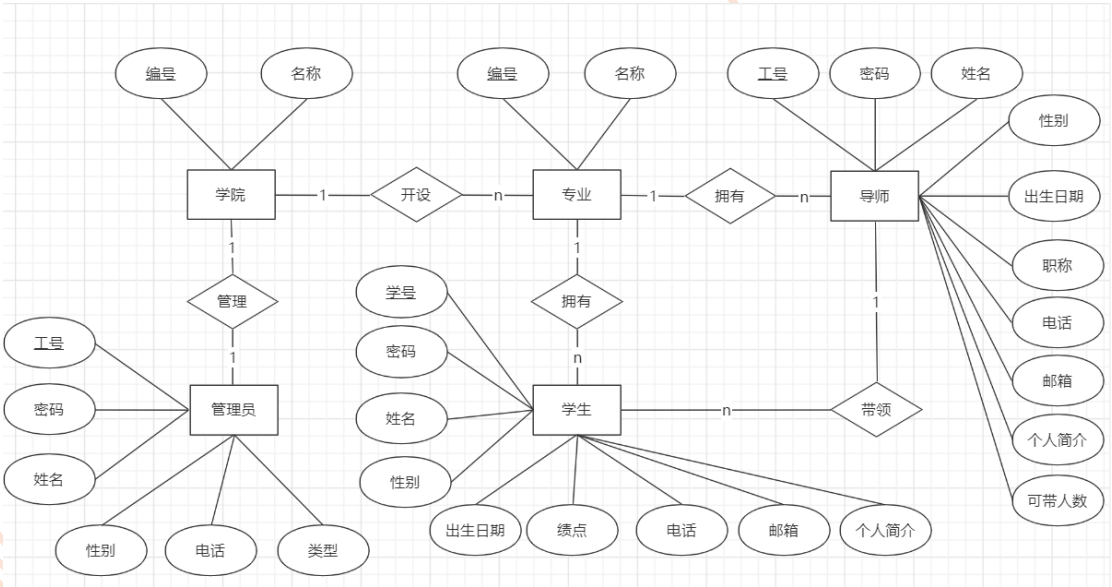

图 4-4-1 系统 E-R 图

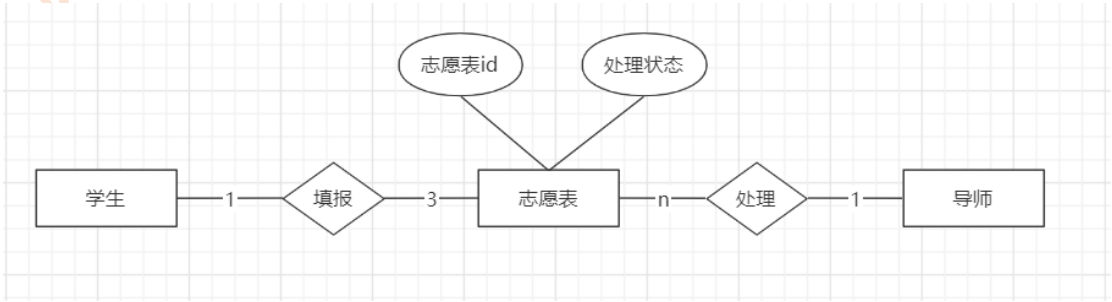

图 4-4-2 学生填报志愿 E-R 图

4.4.2 逻辑结构设计

（1）将 E-R 图转换为关系数据模型

1) 管理员（管理员工号，密码，姓名，性别，电话，类型）；
2) 学院（学院编号，名称，管理员工号）；
3) 专业（专业编号，名称，学院编号）；
4) 导师（导师工号，密码，姓名，性别，出生日期，职称，电话，邮箱，个人简介，可带人数，专业编号）；
5) 学生（学号，密码，姓名，性别，绩点，电话，邮箱，个人简介，专业编号，导师工号）；
6) 志愿表（志愿表 id，学号，导师工号，处理状态）

说明：“___”的字段为主键，“____”为外键。

（2）详细表结构信息

表 1 管理员：admin

字段	类型	允许为 NULL	键	约束	备注信息
ano	varchar(10)	NO	主键		院级管理员工号
password	varchar(20)	NO			密码
aname	varchar(30)	NO			姓名
sex	char(2)	NO		“男”或“女”	性别
tel	char(11)	NO			电话
typeno	int	NO		0 或 1	0 为超管，1 为院级管理员

表 2 学院：dept

字段	类型	允许为 NULL	键	约束	备注信息
dno	varchar(10)	NO	主键		学院编号
dname	varchar(30)	NO			学院名称
ano	varchar(8)	NO	外键		院级管理员编号

表 3 专业：major

字段	类型	允许为 NULL	键	约束	备注信息

mno	varchar(10)	NO	主键		专业编号
mname	varchar(30)	NO			专业名称
dno	varchar(8)	NO	外键		学院编号

表 4 导师：tutor

字段	类型	允许为 NULL	键	约束	备注信息
tno	varchar(10)	NO	主键		导师工号
password	varchar(20)	NO			密码
tname	varchar(30)	NO			姓名
sex	char(2)	NO		“男”或“女”	性别
birth	date	YES			出生日期
title	varchar(5)	NO		“副教授”或“教授”	职称
tel	char(11)	YES			电话
email	varchar(30)	YES			邮箱
description	text	YES			个人简介
ability	int	NO		默认值 3	可带人数
mno	varchar(8)	NO	外键		专业编号

表 5 学生：student

字段	类型	允许为 NULL	键	约束	备注信息
sno	varchar(10)	NO	主键		学号
password	varchar(20)	NO			密码
sname	varchar(30)	NO			姓名
sex	char(2)	NO		“男”或“女”	性别
birth	date	NO			出生日期
gpa	varchar(8)	NO			绩点
tel	char(11)	YES			电话
email	varchar(30)	YES			邮箱
description	text	YES			个人简介
mno	varchar(8)	NO	外键		专业编号
tno	varchar(8)	NO	外键		导师工号

表 6 志愿表：voluntary

字段	类型	允许为 NULL	键	约束	备注信息
sno	varchar(10)	NO			学号
tno	varchar(10)	NO			导师编号
status	int	NO		默认值 0	处理状态 0 表示处理中，1 表示通过，2 表示未通过

表 7 用户登录日志表

字段	类型	允许为 NULL	键	约束	备注信息
account	varchar(10)	NO			登录账号
name	varchar(30)	NO			登录姓名
logtime	datetime	NO			登录时间

4.4.3 物理结构设计

（1）建立数据库结构脚本代码

见附件 create.sql 和 populate.sql。

4.5 系统详细设计与界面设计

4.5.1 登录系统详细设计

（1）设计类如图 4-5-所示。

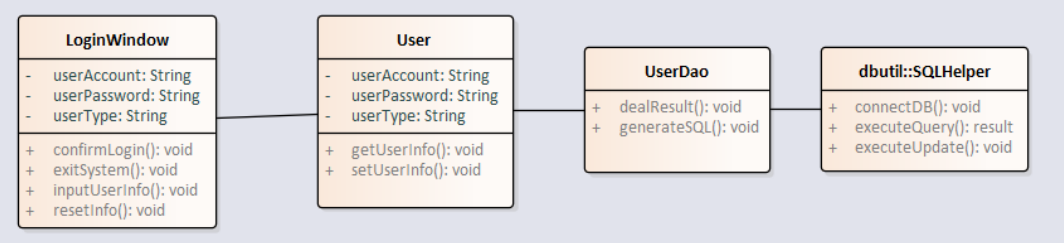

图 4-5-1 登录详细设计类图

LoginWindow 为登录界面类：负责接受用户输入的用户类型，用户账号和用户密码，确认登录、重置信息以及退出系统功能（在实际实现中登录界面是

使用 jsp 即 HTML 编写的，此处的类只是一种代表）。

User 为用户实体类：置有用户账号，用户密码和用户类型属性，使用 getUserInfo()方法获得用户信息。

UserDao 类为控制类：负责写入相应 SQL 语句代码，能够处理返回结果信息。

SQLHelper 类：数据库连接工具类，建立起系统与数据库之间的连接，执行相关的数据库访问操作。executeQuery()完成执行 select 查询语句返回查询的结果集合。excuteUpdate()完成执行更新数据库的 sql 语句。

（2）动态模型，用户登录系统时序图如图 4-5-2 所示。

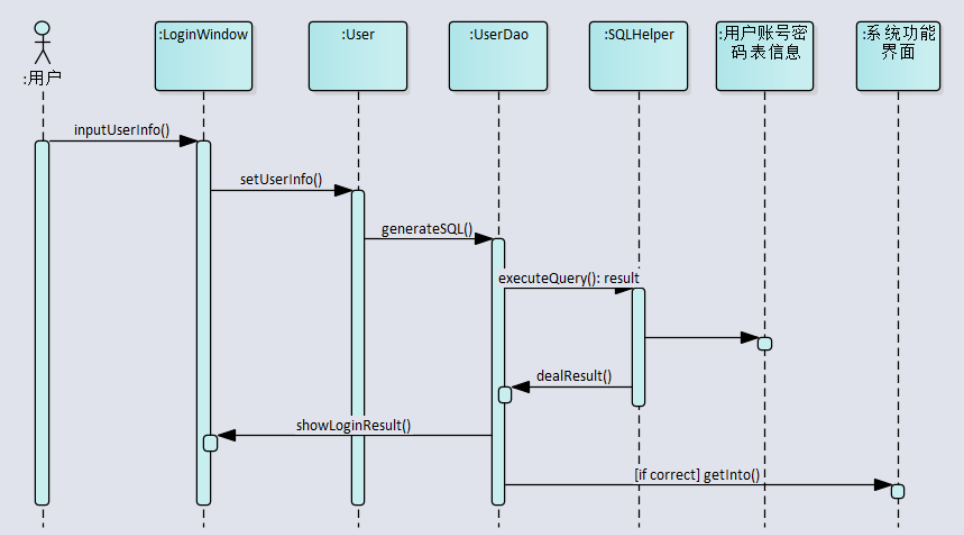

图 4-5-2 用户登录系统时序图

（3）执行功能时相关层次调用，见图 4-5-3。

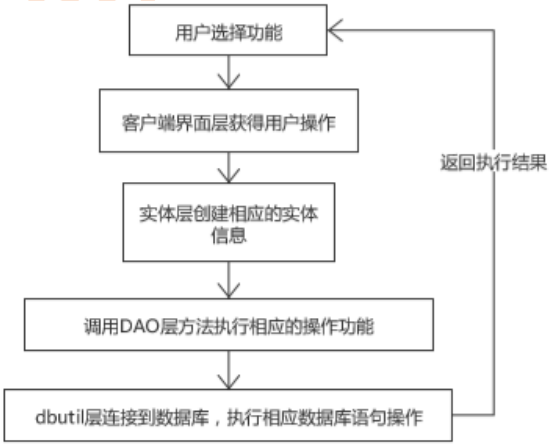

图 4-5-3 功能调用层次图

（4）登录界面 UI

对于登录界面，需要其简洁明了同时具有一定的审美设计，能够让用户对其功能容易上手，其设计界面如图 4-5-4 所示。

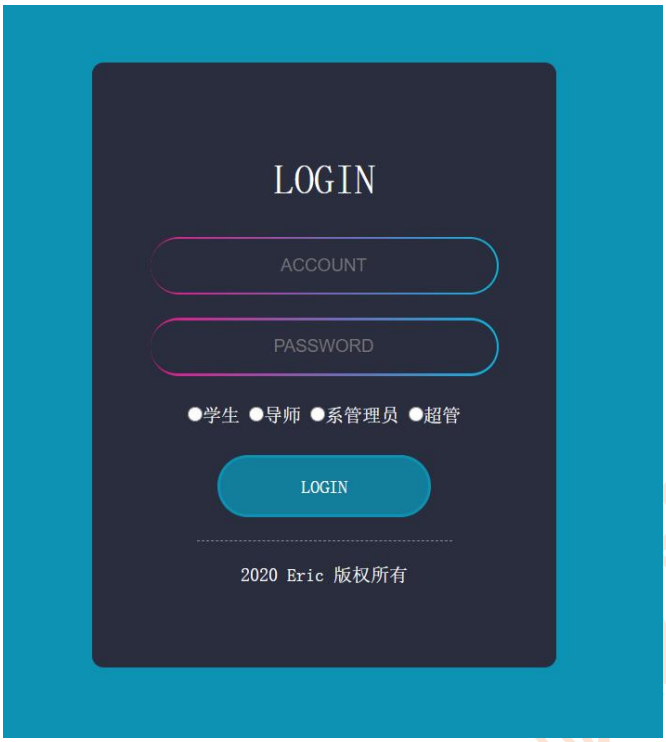

图 4-5-4 系统登录界面

用户需要输入账号和密码，同时应选择自己的身份角色。在本页面中，使用 JavaScript 对输入信息进行校验，如果有空缺信息未选择，则不提交此登录表单，并向用户显示“输入信息均不能为空”提示。当用户所有信息都选择好并点击登录时，此登录表单会提交给控制层来验证用户身份信息的有效性，如果用户信息无效，则返回登录页面重新输入；有效，则进入相应的功能页面。

4.5.2 用户主界面详细设计

所有用户成功登录之后，都会进入相应角色的功能主界面。为了便于界面代码复用，应设计对所有用户的主功能界面使用同样的界面样式框架。用户界面设计图如下 4-5-5 所示。

图 4-5-5 用户功能主界面

对于具体的用户角色，会设计不同的具体功能，但是所有用户的功能页面如上所示。其中区域①显示具体的功能，如果用户选择相应的功能，则其结果

页面显示在区域②，用户在区域②中操作详细的功能权力，同时区域②也是系统的核心模块，使用 iframe 来切换不同的页面。另外，对于不同的用户，在区域③中显示具体的用户名，并可以执行退出系统按钮。

4.5.3 用户详细信息显示页面设计

对于每个用户，都有查看和维护个人信息的功能，同时管理员还可以进行对学生或导师更高级别的信息修改。所以应该设计一个统一的个人信息显示与维护页面。以显示学生信息为例，其设计界面如下图 4-5-6 所示。

图 4-5-6 用户信息显示及维护页面

上述页面对应具体学生的详细，同时对于不同的用户角色，其页面样式也相类似，只是具体信息发生些许变化。同时本页面应设计提供修改信息功能，用户点击修改按钮时，可对相应的信息进行修改。

4.5.4 显示用户信息表格的页面设计

本系统在需求分析阶段发现很多是以表格形式显示多个成员信息的，如：学生查看导师列表、管理员查看管理本系别学生等等，这些功能都需要对多个用户信息进行展示，所以为了更良好的代码结构与代码复用，应处理样式的统一与格式的设计。我们现在以管理员查看学生信息的列表为例，如图 4-5-7 所示。

图 4-5-7 管理员查看学生列表页面

其中区域①应如上设计，可以按照系别对不同的学生进行分类查询（这个功能可以根据不同的角色进行不同的设计），并显示记录的数量。区域②是对应的每一条学生的像信息信息，显示学生的学号姓名等，并且是显示多条记录。

4.5.5 修改密码功能设计

每个用户都拥有修改密码功能，当用户选择修改密码功能时，系统首先要求用户输入账户的原始密码，以确保是本人操作，如图 4-5-8 所示。

图 4-5-8 提示用户输入原始密码

如果原始密码输入有误，则返回不可进入修改密码页面，否则允许用户修改密码。在修改密码时，需要用户提供两次新密码的输入，以防手误（使用 JavaScript 判断两次输入的密码是否一致），如果两次输入密码一致，则更新用户的密码，否则返回。修改密码页面如图 4-5-9 所示。

图 4-5-9 用户修改密码

4.5.6 学生填报志愿和查看志愿功能设计

当学生未确定导师关系之前，可以填报志愿并查看志愿的审核信息，同时需要在系统开放填报志愿的时间段内。在填报志愿时，可供选的导师为同专业导师，且其当前所带人数未满。重要说明一点，所有的志愿为平行志愿，志愿可以三个选项可以相同，且不分优先级，且第一个同意当前学生志愿的导师确定为师生关系，则不再进行剩余志愿。如果志愿被导师拒绝，则会显示拒绝提示。学生可以在志愿填报开放时间里重新填报志愿，并注意重新填报的志愿会覆盖掉上一次的志愿信息。页面设计示例如图 4-5-10，图 4-5-11 所示。

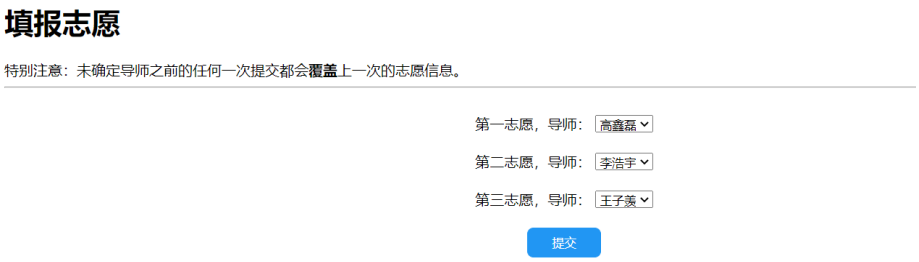

图 4-5-10 填报志愿信息

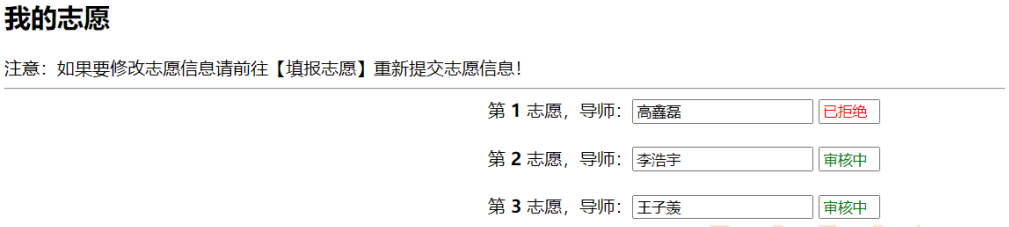

图 4-5-11 查看志愿审核信息

4.5.7 导师处理学生志愿功能设计

导师可以查看当前选择自己的学生志愿信息，能够查看学生详细信息，以此为依据决定是否接收学生的志愿，如果接收，则添加到带领学生的列表中取，如果拒绝，则会通知相应的学生当前的志愿被已被拒绝。并在在此页面可以实时显示当前导师目前可带人数，如果可带人数已满，则不能够再同意其他学生的志愿。查看志愿信息页面如图 4-5-12 所示。

图 4-5-12 导师查看当前的学生志愿

4.5.8 系管理员主要功能设计

（1）学生管理

系管理员可以查看本系别所有学生信息，也可以按照专业显示学生列表，如图 4-5-13 所示。

图 4-5-13 系别下的学生列表

系管理员可以对本系下为分配导师的学生分配导师，但已分配导师的学生不能让管理员再次分配，如果非要分配，管理员可以先强制解除师生关系，然后再进行分配，解除关系见学生-导师关系功能。为未分配导师的学生分配导师页面如图 4-5-14 所示。

图 4-5-14 系管理员为未分配导师的学生分配导师

（2）学生-导师关系管理

系管理员可以查看本系别下的学生-导师关系列表，并能够分类查询或导出当前师生关系表。另外，对于不恰当的师生关系，管理员可以予以解除其关系。设计页面如图 4-5-15 所示。

图 4-5-15 系管理员查看师生关系表

4.5.5 研究生院管理员主要功能设计

超管为负责管理整个研究生院的信息，其功能最强大且复杂，在本系统中，需要为超管设置的功能有：维护个人信息、管理所有学生及导师信息、设定填报志愿开放时间、查看或导出学生导师分配结果、重置用户密码、导入学生及导师信息（单个导入以及批量导入）。

（1）设定志愿开放时间

学生填报志愿页面要求只有在一定的时间段才允许进入，即需要设计填报志愿的开放时间，这一权限本系统设计在了超管身上。超级管理设定志愿开放时间如图 4-5-16 所示。

图 4-5-16 设定志愿开放时间

（2）重置用户密码

为了避免用户忘记密码后难以找回，本系统在超管功能中设计重置用户密码功能，此功能是根据账号类型：学生、导师和管理员，来对相应类型的用户进行密码重置，重置后的密码与用户的账号保持一致，用户可以利用此密码登录系统后重新设定密码。其页面设计如图 4-5-17 所示。

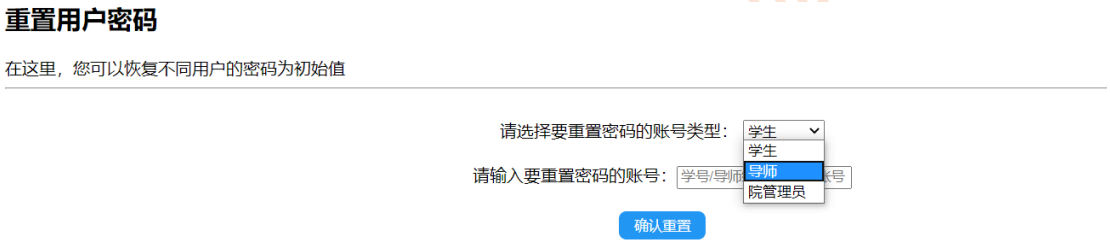

图 4-5-17 重置用户密码

（3）单个导入（以学生为例，单个导入导师信息与此类似）

在一些特别情况下，需要导入单个学生。超管可以单个导入学生，其设计页面如图 4-5-18 所示。其中，本系统需要设计在导入单个学生信息时的必填项目，用符号“*”标记。确定录入后，系统设计为提示录入成功信息，同时向数据库中的学生表当中插入学生信息。

图 4-5-18 单个导入学生

（4）批量导入（以学生为例，批量导入导师信息与此类似）

超管可以使用.xlsx 表格批量导入学生或导师信息，对于导入表格的结构信息，本系统需要设定一定的要求限制，例如对于批量导入学生信息，其上传说明以及页面设计如图 4-5-19 所示。

学号	姓名	性别	出生日期	绩点	电话	邮箱	专业名称
20171412	张三	男	2000-01-01	2.532	12345678912	20171412@qq.com	应用物理
20171413	王红	女	1999-12-31	2.456	12345678913	20171413@qq.com	英语
20171414	李四	男	1999-02-28	1.586	12345678914	20171414@qq.com	法语

图 4-5-19 批量导入学生信息

五、系统核心模块描述

5.1 使用 PreparedStatement 执行可变参数的更新语句

Java 提供了 Statement、PreparedStatement 和 CallableStatement 三种方式来执行查询语句，其中 Statement 用于通用查询， PreparedStatement 一般用来执行带 IN 参数或不带参数的查询语句，而 CallableStatement 则是用于存储过程。其结构层次图如 5-1-1 所示。

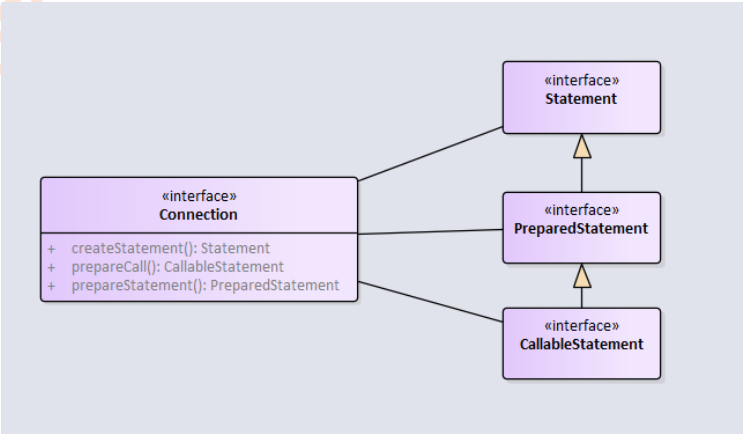

图 5-1-1 Java 提供的查询语句结构图

PreparedStatement 是 java.sql 包下面的一个接口，继承了 Statement，并与之

在两方面有所不同。通过调用 connection.preparedStatement(sql)方法可以获得 PreparedStatment 对象。数据库系统会对 sql 语句进行预编译处理，预处理语句将被预先编译好，这条预编译的 sql 查询语句能在将来的查询中重用，因此它比 Statement 对象生成的查询速度更快。

在本次系统详细设计中，使用 PreparedStatement 的方法源代码如下：

```
public static boolean executeUpdate(String mysql, Object... params) {
    buildConnection();
    int r = 0;
    try {
        pre = conn.prepareStatement(mysql);
        int count = 1;
        for (Object param : params) {
            if (param instanceof String) {
                pre.setString(count++, param.toString());
            } else if (param instanceof Integer) {
                pre.setInt(count++, new Integer(param.toString()));
            } else if (param instanceof Float) {
                pre.setFloat(count++, new Float(param.toString()));
            } else {
                pre.setObject(count++, param);
            }
        }
        r = pre.executeUpdate();
    } catch (Exception ex) {
        ex.printStackTrace();
    } finally {
        closeConnection();  // 关闭连接
    }
    return r != 0;
}
```

选择使用 PreparedStatement 的原因有以下几点：

（1）PreparedStatement 可以写动态参数化的查询

可以使用占位符。?为占位符，一个占位符只允许有且仅有一个值（所以这也是 PreparedStatement 的一点局限性）。设置参数值（占位符值）时要根据参数的类型选择不同的 set 方法，值得注意的是：占位符的索引位置从 1 开始而不是 0，如果填入 0 会导致 java.sql.SQLException invalid column index 异常。

（2）PreparedStatement 可读性和维护性更强

正如上述代码所示，PreparedStatement 可以使用参数进行执行 sql 语句，所以在一些复杂执行语句情况下，使用 PreparedStatement 对象可以更清晰的

阅读与维护代码。

（3）PreparedStatement 可以提高性能

因为预编译语句有可能被重复调用，所以预编译语句在被 DB 编译器编译后的执行代码会被缓存下来，那么下次调用时只要是相同的预编译语句就不需要编译，只要将参数直接传入编译过的语句执行代码中(相当于一个函数)就会得到执行。这并不是说只有一个 Connection 中多次执行的预编译语句被缓存，而是对于整个 DB 中，只要预编译的语句语法和缓存中匹配，那么在任何时候就可以不需要再次编译而可以直接执行，所以它的性能更高。而 statement 的语句中，即使是相同一操作，而由于每次操作的数据不同所以使整个语句相匹配的机会极小，几乎不太可能匹配。

（4）PreparedStatement 更安全

使用 PreparedStatement 的参数化的查询可以阻止大部分的 SQL 注入。在使用参数化查询的情况下，数据库系统不会将参数的内容视为 SQL 指令的一部分来处理，而是在数据库完成 SQL 指令的编译后，才套用参数运行，因此就算参数中含有破坏性的指令，也不会被数据库所运行。

5.2 登录验证

当用户点击登录时，首先要利用 JavaScript 来进行输入信息的检验，检查是否有空项，如果有空项则不提交登录表单，否则进行跳转。

```
// 验证用户登录时输入的信息是否填写完整
function submitForm() {
    // 获得账号
    let account = document.getElementsByName("account")[0].value;
    // 获得密码
    let password = document.getElementsByName("password")[0].value;
    // 获得用户类型
    let typeArray = document.getElementsByName("type");
    // 获得选中的用户类型
    let typeValue = null;
    for (let i = 0; i < typeArray.length; i++) {
        if (typeArray[i].checked) {
            typeValue = typeArray[i].value;
        }
    }
    // 判断是否有空输入，如果无空输入，则提交登录表单
    if (account != null && password != null && typeValue != null) {
        let form = document.getElementById('loginForm');
        form.submit();
```

```
    } else {
        alert("输入均不能为空！")
    }
}
```

当提交登录表单之后，就由控制层来验证用户信息是否有效，如果无效，则返回登录页面，并提示用户登录失败信息；如果有效，则登录进相应的用户功能页面。以学生为例，当验证学生信息有效时，其处理源代码如下：

```
protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    HttpSession session = request.getSession();
    String sno = (String) session.getAttribute("account");
    String type = (String) session.getAttribute("type");
    //  把学生、专业、系别、导师实体存储到会话
    Student student = StudentDAO.getInstance().getStudentBySno(sno);
    session.setAttribute("student", student);
    Major                              major                              =
MajorDAO.getInstance().getMajorByMno(student.getMno());
    session.setAttribute("major", major);
    Dept dept = DeptDAO.getInstance().getDeptByDno(major.getDno());
    session.setAttribute("dept", dept);
    if (student.getTno() != null) {
        Tutor                                  tutor                          =
TutorDAO.getInstance().getTutorByTno(student.getTno());
        session.setAttribute("tutor", tutor);
    }
    // 定向到学生页面
    response.sendRedirect("student.jsp");
}
```

5.3 切换用户功能

本系统把每个用户各项功能的显示页面存放在一个 iframe 里，其源代码如下：

```
<div id="middle">
    <div id="container">
        <table style="margin: 0 auto;">
            <iframe  src="welcome.jsp"  id="myIframe"  frameborder="0"
width="100%" height="100%">
            </iframe>
        </table>
    </div>
</div>
```

当用户点击左侧导航栏功能选项时，就会触发一个名为 changeFunction 的

JavaScript 函数，例如：学生用户点击个人信息功能，其源代码如下：

```
<div class="function"
onclick="changeFunction('student_personalInfo.jsp')">
        个人信息
</div>
```

此时就会触发 changeFunction 函数，并向此函数传递一个 url 作为参数，此时在 changeFunction 函数内部则会重新设置 iframe 的 url，使之显示指定页面，从而达到不同的功能按钮显示不同信息页面的功能。设置 iframe 的 url 的 JavaScript 源代码：

```
// 获得功能页面的框架
let iframe = document.getElementById('myIframe');
// 设置功能页面要显示内容的 url
function changeFunction(url) {
    iframe.setAttribute('src', url);
}
```

六、总结

6.1 系统已经实现的功能及可扩充的方向

6.1.1 系统实现的用户功能

（1）学生端

1）维护个人基本信息；

2）查看专业导师信息；

3）填报志愿与查看志愿审核状态。

（2）导师端

1）维护个人基本信息；

2）查看带领的学生的信息；

3）查看选我的志愿信息。

（3）系负责人端

1）维护个人信息；

2）管理学生和导师信息（可按照专业查找学生或导师信息）；

3）查看师生关系表并可以执行解除操作。

（4）研究生院负责人端

1）维护个人信息；

2）管理学生和导师信息（可按照学院查找学生或导师）；

3）设定填报志愿的时间；

4）查看师生关系表（仅提供查看功能）并可以导出；

5）重置用户密码；

6）对学生和导师信息的录入。实现了对学生或导师的单个录入以及批量录入。

6.1.2 可以扩充的方向

（1）表格分页功能

本系统在设计每个显示用户信息的列表时，未设计表格的分页功能，这样如果数据量较大的话，会给用户不好的体验，所以可以给每个显示列表的页面增加表格分页的功能，并支持跳转指定页号的操作。

（2）增加搜索框

目前学生用户在浏览本专业的导师时，只能从上到下依次的查看导师，会带给用户不好的感受，为了提升用户满意度以及功能上的需求，应增加对导师的搜索框，比如按照导师姓名进行搜索等。这一点对于导师端同样适用。

对于管理员端，因为要面对更多的信息数据，所以更应该设计合理的不同的搜索粒度来查找相应的学生或导师信息。

（3）设计信息的联动变化

对于研究生院管理员，其管理的是各个学院以及学院下的专业信息，本系统应一项功能，实现学院和专业的联动变化。例如，当研究生院管理员单个录入学生时，可以先选择此学生的学院，而此时专业名称跟随学院名称做出相应的变化。这一功能在研究生院管理员查找学生或导师信息也会给用户以良好的体验。

（4）设置志愿的优先级

在本系统的设计中，在学生填报志愿功能中设置的为平行志愿，而更合理的设计应该是按照志愿的先后顺序设定志愿的优先级。并且对于优先级高的志愿，相应的导师可以优先进行选择操作，只有他拒绝之后，下一优先级的志愿信息才会发送给第二个导师，供其选择，否则志愿不往下继续发送。

（5）添加智能分配功能

在学生填报志愿以及导师接收志愿的过程中，不免发生学生选择同一导师或导师选择同一学生的情况，此时会剩余大量的导师以及学生未完成分配，如果要管理员进行一一分配，在大数据量的情况下无疑是一场复杂与耗时的工作，所以可以为系统增加只能分配导师功能。

此功能的主要作用：采用算法，结合学生的绩点、排名、志愿顺序等以及导师的学生人数等信息，对剩余未分配到导师的学生进行智能分配。

6.2 遇到的问题及其解决方法

（1）IDEA 系统配置与项目部署问题

在一开始使用 IDEA 创建 Web 项目并部署运行时，在浏览器地址栏中输入相应的 url 请求并不能显示页面，最终找到原因是项目没有部署到 tomcat 目录下的 webapps 当中，所以导致了程序不能运行。所以经过查询一些博客与资料，最终学会了在 IDEA 中配置 tomcat 的操作。具体的步骤如下：

首先 IDEA 中的 Web 项目中 WEB-INF 目录下没有 classes 与 lib 文件夹，需要自己创建，如图 6-2-1 所示。

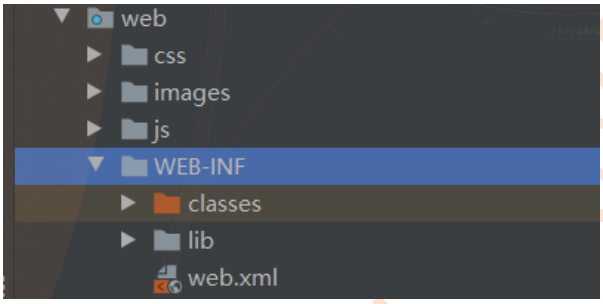

图 6-2-1 在 WEB-INF 下创建 classes 与 lib 文件夹

在创建好文件夹之后，需要进行此目录的路径设置，即把项目的字节码文件与依赖的 jar 包路径改为这两个路径。其做法是选择项目结构中的 Modules 选项，之后选择 Paths，然后设置两个输出路径为刚才创建的 classes 目录。如图 6-2-2 所示。

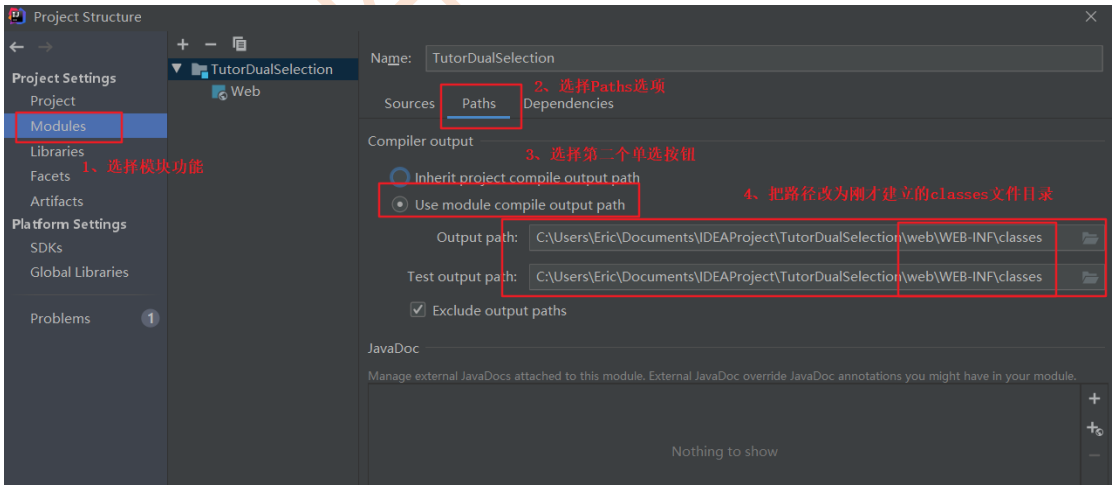

图 6-2-2 设置导出的 classes 文件目录

在设置完 classes 之后，还需要对 lib 目录进行配置，同样是在 Modules 中，此时切换到 Dependencies 选项，点击+号然后选择 JARs 选项，把刚才创建的 lib 目录添加进来即可。如图 6-2-3 所示。

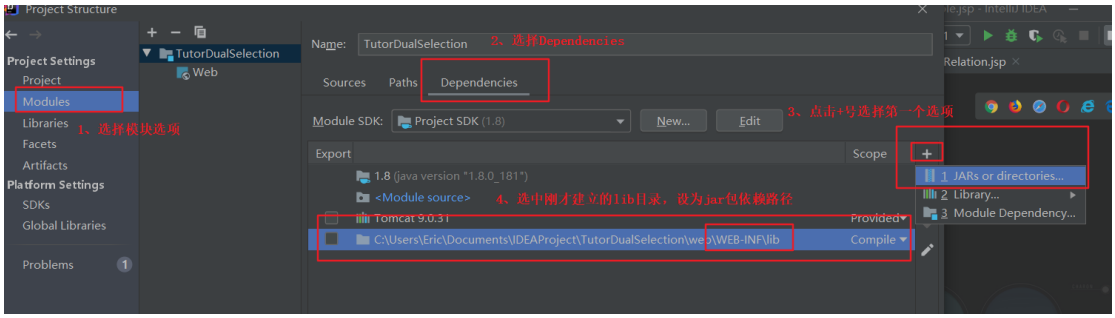

图 6-2-3 配置 lib 目录

然后选择 Artifacts 选项，并把 Output directory 中的路径改为 tomcat 下的 webapps 文件，这一配置是将此项目部署到 webapps 文件夹中去，配置如图 6-2-4 所示。

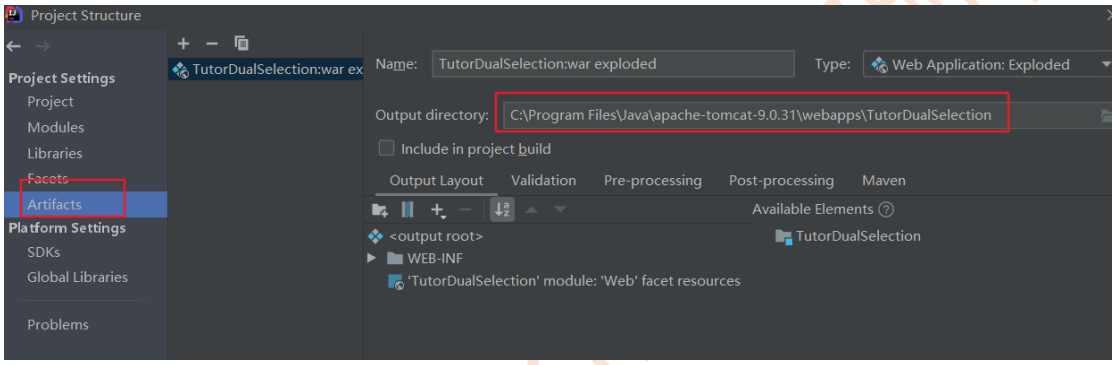

图 6-2-4 将项目部署到 webapps 文件夹下

接下来是配置 IDEA 实现项目的热部署，即在修改了页面信息之后可以实时部署到服务器并刷新浏览器即可显示新的页面功能，打开 tomcat 的配置页面，选择图 6-2-5 中红色方框里的内容，点击确认即可。

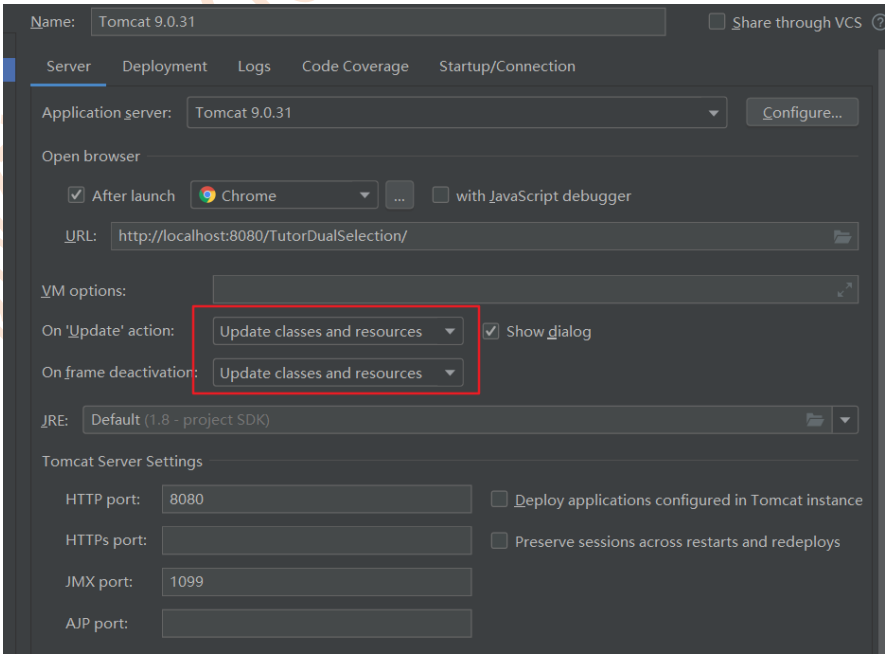

图 6-2-5 配置 tomcat 的热部署

综上配置，在 IDEA 中的 tomcat 算是配置完成，并能够运行起来。

（2）页面复用问题

在本系统的开发过程中，会有许多相似的页面功能，比如查看个人信息、修改个人密码等。在开发初期，为了尽快完成相关功能，并没有对页面进行整合与复用，而是基本上每个功能都编写相应的界面，导致最后的代码管理起来比较困难，而且 jsp 文件较多。为了减少一些冗余代码，减少代码量，并减少编写的 jsp 文件，必须要对这些相似的功能进行合理的设计，以达到其他用户功能可以直接复用的目的。所以在后期开发过程中对相似的功能页面进行了整合，使用控制层来通过传递的参数来决定页面的一些 session 值，从而使同一个页面文件可以根据需求动态的显示内容。在本系统中，同样实现了前后端的分离，使得系统更好维护。

（3）导入 hutool-all-4.3.1.jar 失败的问题

在系统最后为超管添加表格导入导出信息的时候，使用到了依赖 jar 包 hutool-all-4.3.1.jar，但是在导入此 jar 包时却出现了问题：在把此 jar 包复制到 WEB-INF 下的 lib 目录里后，并不能显示此 jar 包下的目录，即此时不能使用 jar 中的相应类和文件。当出现这个问题后，我首先将 jar 移除然后重新导入，发现还是不行，最后，我想是不是在整个项目依赖的 jar 文件中也需要导入此 jar 包，所以我就在整个项目的 External Libraries 中导入了 hutool-all-4.3.1.jar 文件，神奇的是，当在整个项目依赖中导入此包，在 WEB-INF 下 lib 中的 hutool-all-4.3.1.jar 竟然也显示出了二级目录，此时代表可以 jar 包成功导入并且可以使用其功能类。并且最后实践证明，整个项目的依赖中并不需要导入此 jar 包，只需复制在 WEB-INF 下的 lib 文件夹中即可。至于一开始为什么复制不成功，可能是由于项目没有正确加载对此包的依赖关系，但是其他的 jar 包文件都是可以正确复制到 lib 目录下。

6.3 开发过程中学到的技术

（1）MVC 开发模式

MVC 模式一直都有听说，也理解其含义，即把软件开发层次分为模型层、控制层和视图层，但是一直没有自己动手实践过。在前期的课程学习中，并没有使用前端的一些语言如 HTML 等进行开发，所以对其的具体实现并不了解，而通过这次课程的学习以及本次系统的设计，可以说是对 MVC 模式有了更深一步的理解。

JSP 在 HTML 中加入了大量的、复杂的业务逻辑，如果后期业务逻辑发生改变，修改 JSP 就会捉襟见肘；Servlet 虽然解决了业务逻辑的问题，但是通过字符串拼接的方式生成动态的 HTML 页面，也会导致代码臃肿，难以维护；

MVC(Model-View-Controller) 模式就扬长避短，将两者完美结合在一起，它把把软件系统分为三个层次：模型 Model、视图 View 和控制器 Controller，通常 JSP 为 View 层，Servlet 为 Controller 层，JavaBean 为 Model 层，其运行过程如下图 6-3-1 所示。

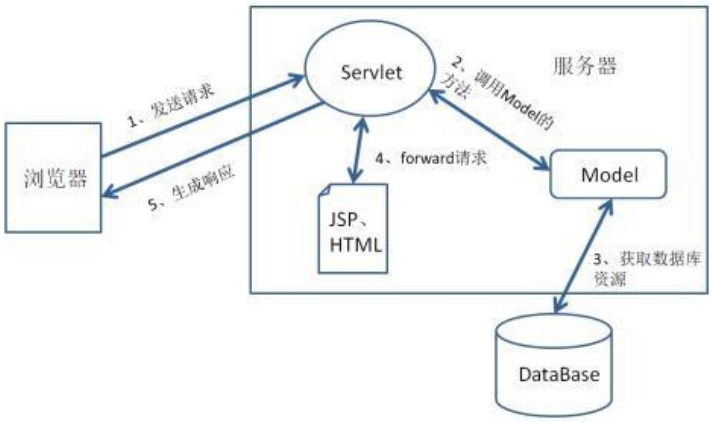

图 6-3-1 MVC 模式的层次调用

（2）导入导出文件与 Excel 表的解析

在通过本次系统开发过程中，遇到的唯一比较陌生的功能就是使用代码对文件的操作以及对 Excel 表格的导入与导出功能。在经过不断查找资料与视频的学习，最终解决了文件上传与 Excel 表格的解析并把数据导入进数据库。

文件上传主要使用了 commons-fileupload-1.3.2.jar 和 commons-io-2.5.jar 这两个 jar 包下的类：

org.apache.commons.fileupload.FileItem;

org.apache.commons.fileupload.disk.DiskFileItemFactory;

org.apache.commons.fileupload.servlet.ServletFileUpload;

在解析 Excel 表时，则使用到了 hutool-all-4.3.1.jar 下的两个类：

cn.hutool.poi.excel.ExcelReader;

cn.hutool.poi.excel.ExcelUtil;

具体的实现代码请查看源代码中控制层 controller 包下的 UploadServlet.java 、ParseExcelServlet.java 和 ExportExcelServlet.java 文件。

（3）HTML、CSS 以及 JavaScript 的熟练使用

本系统使用到了大量的.jsp 文件，而作为呈现给用户的视图层，除了注重其实际功能外，更要注重它的审美设计，所以就要求熟练的使用 CSS，对一些常用的设计进行美化样式处理；另外还有一些用户的操作是直接就可以在浏览器中处理的，并不需要向服务器发出请求处理，所以此时起作用的是 JavaScript。通过这三者的结合与合理的设计，最终才能呈现给用户功能齐全并且界面简洁清爽的系统。