华北科技学院计算机学院综合性实验

实 验 报 告

课程名称 Oracle数据库

实验学期 2024 至 2025 学年 第 1 学期

学生所在院部 计算机学院

年级 2022 专业班级 信管B221

学生姓名 李智文 学号 202207034116

成绩评定：

1、工作量： A（ ），B（ ），C（ ），D（ ），F（ ）

2、难易度： A（ ），B（ ），C（ ），D（ ），F（ ）

3、答辩情况：

基本操作： A（ ），B（ ），C（ ），D（ ），F（ ）

代码理解： A（ ），B（ ），C（ ），D（ ），F（ ）

4、报告规范度： A（ ），B（ ），C（ ），D（ ），F（ ）

5、学习态度： A（ ），B（ ），C（ ），D（ ），F（ ）

总评成绩：

任课教师：郭慧

《Oracle 数据库》课程综合性实验报告

开课实验室： 基础七 2024 年 11 月 9 日

实验题目	基于 Oracle 数据库的学生信息管理系统开发

一、实验的目的及要求

1.1 实验目的

掌握 Oracle 数据库的对象的创建和管理方法，巩固对 Oracle 数据库基本原理和基础理论的理解。掌握开发 Oracle 数据库应用程序的技术，巩固对 Oracle 数据库基本原理和基础理论的理解。同时学习使用 jdbc 等技术对 oracle 数据库进行连接和增删改查操作,制作 web 程序

1.2 实验要求

使用 JSP 作为前台,后台连接数据库,完成一个信息管理系统的开发,在 oracle 数据库中新建用户、新建表空间，所有数据库对象都由新用户创建并存储在新建的表空间中编写程序实现对数据库的增删改查等操作,并在前台页面展示结果，用户可在前端操作数据库数据。

二、设备与环境

开发语言：前端语言：vue 框架，html，css，JavaScript

后端语言：java，ssm 框架，jsp

数据库：oracle 数据库，使用 sql 进行数据操作

开发工具：idea 编译器

部署工具：tomcat10，maven 工程

三、实验内容

3.1 系统概述

3.1.1 需求分析

需要实现一个用户登录注册的页面，在用户成功登录后显示学生信息操作的页面。登录页面要求有在登录时选择记住用户名与密码的功能，注册页面要求密码设置规范，验证码验证功能。在学生信息操作页面除了基本的增删改查之外，还需要增加相关对登录用户的操作，修改用户和修改密码等。除此之外页面的学生信息展示要求具有分页功能，包括每页展示信息条数，批量删除用户信息等功能。

3.1.2 系统功能介绍

用户登录界面：用户点击记住我可设置 cookie，下次进入登录界面自动填写。

图 1：用户登录

用户注册界面：给用户名和密码框分别设置了失去焦点事件，一个是通过异步请求查询用户名是否存在，一个是匹配正则表达式规范密码格式。

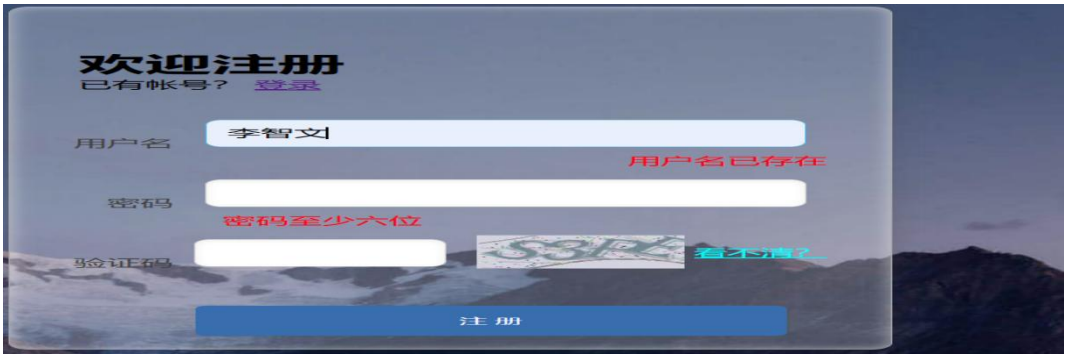

图 2：用户注册

用户操作信息界面：页面除了实现了基本的增删改查外，还有一些功能如下

批量删除功能：每条数据前都有复选框，当我选中多条数据并点击 批量删除 按钮后，会发送请求到后端并删除数据库中指定的多条数据。

分页查询功能：当数据库中有很多数据时，我们不可能将所有的数据展示在一页里，这个时候就需要分页展示数据。

条件查询功能：数据库量大的时候，我们就需要精确的查询一些想看到的数据，这个时候就需要通过条件查询。

在用户要进行增删改操作时会调出相应的输入框（平时隐藏）

鼠标选中可进行批量删除。

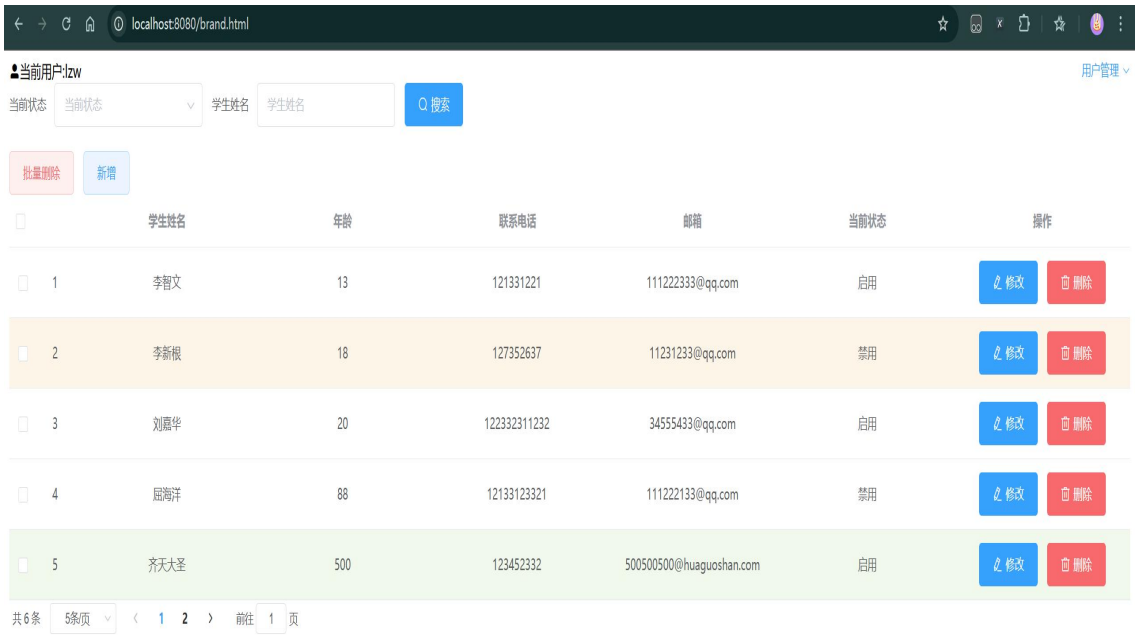

图 3：学生信息管理图

3.1.3 功能模块划分

基于 ssm 框架的 web 项目开发下，我们将程序开发分为四个模块，分别是持久层 dao，业务层 service，控制层 controller 和数据视图层 vive。

持久层使用了 mybatis 框架，主要负责与数据库的连接，sql 语句的执行等操作，同时向数据库提交和接收数据（mapper 的配置文件在 resource 包下统一管理），接收和传递的数据都是从 pojo 包（封装数据模型）中，pojo 包中的类统一分配了构造方法保存数据。MyBatis 的动态 SQL 提供了灵活的数据查询和操

作方式，使数据的插入、更新、删除等操作高效而稳定。

业务层负责系统的核心逻辑处理，作为控制层和持久层之间的中间桥梁。业务层在接收到控制层的请求后，调用持久层的方法执行数据库操作，并根据需求对数据进行逻辑处理后返回给控制层。通过这种方式，业务逻辑实现了对底层数据访问的封装，同时提高了代码的复用性和模块化。业务层的接口设计规范了数据的流转，确保了各模块之间的解耦，有助于系统的扩展和后期维护。

控制层基于 Spring MVC 框架实现，与数据视图层交互并负责请求的分发和响应。控制层的核心职责包括接受前端传来的请求参数，将参数封装为 Java 对象，通过调用业务层服务完成相关操作，然后将处理后的数据返回给前端。Spring MVC 提供了很多注解支持，使得控制层的开发更加简洁、高效。控制层的各个请求路径、参数映射、异常处理均可以直接用注解代替

数据视图层采用 AJAX 技术实现页面的动态渲染和数据的异步加载。通过 AJAX 请求，前端可以在不刷新页面的情况下与服务器进行数据交互，极大提升了用户体验。前端的数据请求发送到控制层，控制层处理完后将数据以 JSON 格式返回给前端，再由 vue 框架 处理和渲染。AJAX 方式的使用使视图层更具交互性,不用频繁地刷新网页就能和后端交换数据

此外，我在项目中使用了 Maven 管理和构建 Java 项目，项目中使用的依赖和 jar 包全都在 pom 配置文件中进行配置。代码交给 git 托管可以更方便后续业务的扩展和功能的改进

3.1.4 项目层次结构

以下是我项目的包层次结构;

后端部分 (java 目录下):

com.lzw 是项目的基础包名

config: 配置包，通常包含 Spring、MyBatis 等框架的配置类

controller: 控制层，处理前端请求，包含登录注册和学生管理的接口

mapper: MyBatis 的数据访问层接口，负责数据库操作

pojo: 实体类包，存放 Student、User 等模型类

service: 业务逻辑层，处理具体业务实现

util: 工具类包，可能包含一些通用方法

资源文件 (resources 目录下):

com.lzw.mapper: MyBatis 映射文件目录

StudentMapper.xml: 学生相关 SQL 映射

UserMapper.xml: 用户(登录注册)相关 SQL 映射

mybatis-config.xml: MyBatis 核心配置文件

前端部分 (webapp 目录下):

css: 样式文件目录

element-ui: Element UI 框架文件

imgs: 图片资源目录

js: JavaScript 文件目录

WEB-INF: Web 应用配置目录

brand.html: 学生信息展示与操作页面

login.jsp: 登录页面

register.jsp: 注册页面

vue.html: Vue.js 相关页面

功能模块划分：

用户认证模块：

登录注册功能

涉及文件：login.jsp, register.jsp, UserMapper.xml

学生信息管理模块：

学生信息的 CRUD 操作

涉及文件：StudentMapper.xml, 相关的 Controller 和 Service 实现

这是一个基于 SSM(Spring + SpringMVC + MyBatis)框架的学生信息管理系统，采用前后端分离架构。后端采用分层设计，包含 controller 控制层处理请求，service 业务层实现逻辑，mapper 数据访问层操作数据库，以及 pojo 存放实体类。前端使用 Vue.js 结合 Element UI 构建用户界面，通过 JSP 和 HTML 展示页面。系统主要分为两大功能模块：用户认证模块(包含登录注册)和学生信息管理模块(实现学生信息的增删改查)。项目配置文件存放在 config 包和 resources 目录下，

包括 Spring 和 MyBatis 的相关配置。

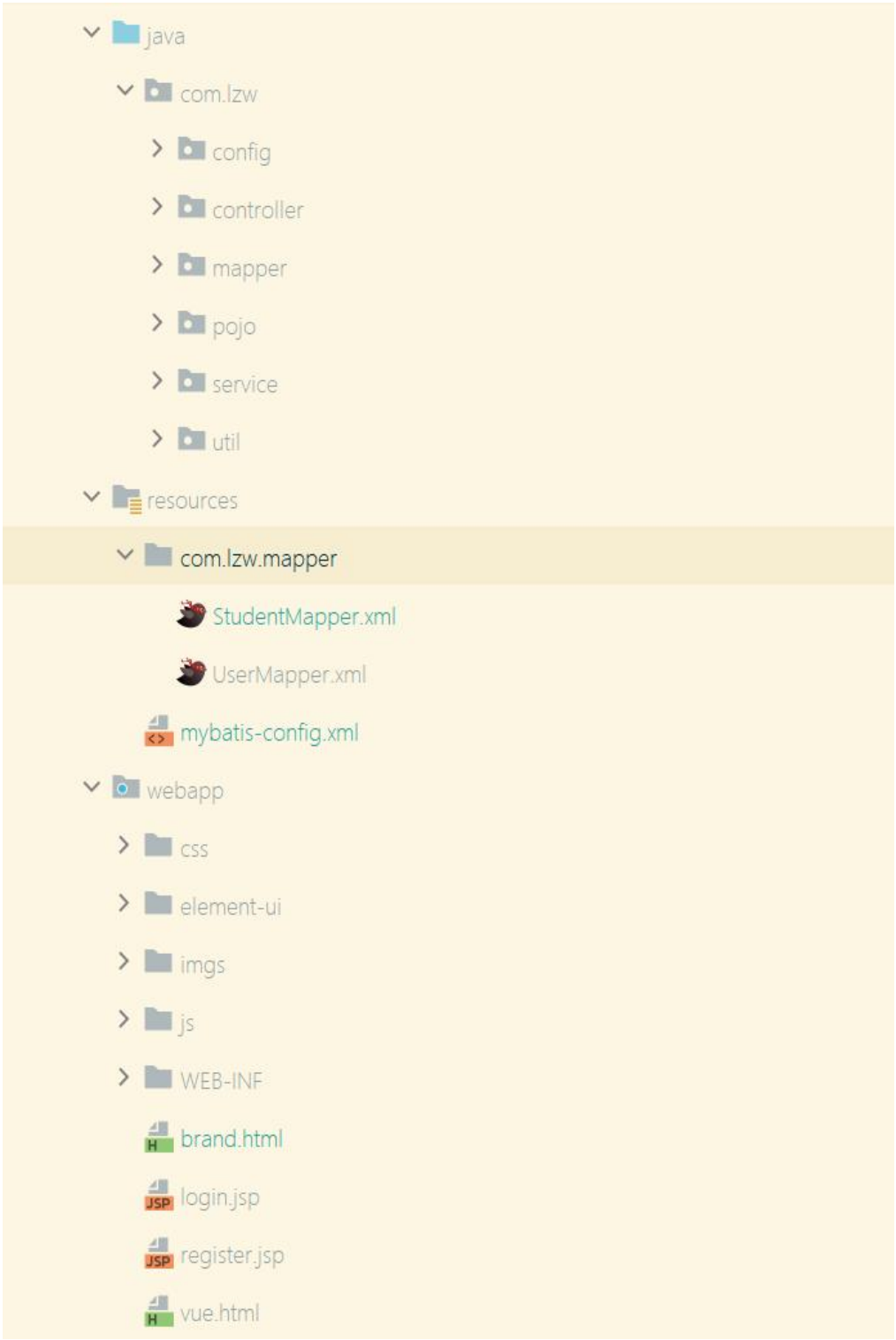

图 4:项目业务层次表

3.2 设计应用数据库

在设计数据库时，考虑到项目的扩展性和维护需求，我对表的结构进行了初步规划。两个核心表分别为用户表 tb_user 和学生信息表 stu。tb_user 表主要用于记录操作人员的登录信息，包括用户名和密码，以确保项目的安全性和用户数据的隐私。stu 表则用于管理学生的基本信息，涵盖了姓名、联系方式、年龄和当前状态等字段。

为了保证数据的完整性，两个表均使用了 ID 自增字段作为主键，避免了手动设置的复杂性。此外，两个表都指定在表空间 work1 中，以便数据集中管理，有助于提高查询和备份的效率。

未来在项目的开发中，可能会进一步扩展数据库设计。例如，可以在 tb_user 表中添加角色字段（如 role），实现不同权限用户的区分管理；在 stu 表中增加入学时间等字段，以便实现更复杂的学生数据分析。同时，对于 stu 表中的 status 字段，可以定义状态码（如 0 表示在校，1 表示毕业等）用于状态的快速识别与筛选。

以下是创建两个表的建表语句

```
CREATE TABLE lzw.tb_user(
    id NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY,
    username VARCHAR2(50),
    password VARCHAR2(50)
)TABLESPACE work1;
CREATE TABLE LZW.stu (
    id NUMBER GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY,
    name VARCHAR2(100),
    phone VARCHAR2(20),
    email VARCHAR2(100),
    age NUMBER,
    status NUMBER
)TABLESPACE work1;
```

以下是创建完两个表之后的表结构

列 | 数据 | 约束条件 | 授权 | 统计信息 | 触发器 | 闪回 | 相关性 | 详细资料 | 分区 | 索引 | SQL

操作...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ID	NUMBER	No	"LZW"."ISEQ$$_73405".nextval	1	(null)
2	NAME	VARCHAR2(100 BYTE)	Yes	(null)	2	(null)
3	PHONE	VARCHAR2(20 BYTE)	Yes	(null)	3	(null)
4	EMAIL	VARCHAR2(100 BYTE)	Yes	(null)	4	(null)
5	AGE	NUMBER	Yes	(null)	5	(null)
6	STATUS	NUMBER	Yes	(null)	6	(null)

表 1：学生信息表 student

列 | 数据 | 约束条件 | 授权 | 统计信息 | 触发器 | 闪回 | 相关性 | 详细资料 | 分区 | 索引 | SQL

操作...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ID	NUMBER	No	"LZW"."ISEQ$$_73402".nextval	1	(null)
2	USERNAME	VARCHAR2(50 BYTE)	Yes	(null)	2	(null)
3	PASSWORD	VARCHAR2(50 BYTE)	Yes	(null)	3	(null)

表 2：用户信息表 tb_user

3.3 开发应用程序

3.3.1 数据库连接层

在持久层我使用了 mybatis 来封装 jdbc 对数据库进行连接，首先需要在 pom 文件中引入 ojdbc8 和 mybatis 的包依赖。引入依赖之后需要在 mybatis 的配置文件 mybatis-config.xml 文件中配置数据库驱动和数据库连接相关的信息

```
1.  <environment id="development">
2.          <transactionManager type="JDBC"/>
3.          <dataSource type="POOLED">
4.              <property name="driver" value="oracle.jdbc.driver.OracleDriver"/>
5.              <property name="url" value="jdbc:oracle:thin:@localhost:1521:ORCL"/>
6.              <property name="username" value="LZW"/>
7.              <property name="password" value="123456"/>
8.          </dataSource>
9.      </environment>
```

<transactionManager type="JDBC"/>：指定事务管理器类型为 JDBC，这意味着事务管理将由 JDBC 直接控制，不需要容器管理的事务。

<dataSource type="POOLED">：定义数据源的类型为 POOLED，即连接池模式。通过连接池复用连接资源，减少连接数据库的时间和系统开销。

<property name="driver" value="oracle.jdbc.driver.OracleDriver"/>：配置数据

库驱动为 OracleDriver，这是 Oracle 数据库的 JDBC 驱动，用于建立 Java 与 Oracle 数据库的连接

<propertyname="url"value="jdbc:oracle:thin:@localhost:1521:ORCL"/>：指定连接 URL，其中：

jdbc:oracle:thin 表示使用 Oracle 的 Thin 驱动。

@localhost:1521 指定数据库服务器的 IP 和端口（这里是本地 1521 端口）。

ORCL 表示 Oracle 数据库实例的 SID 名称。

<property name="username" value="LZW"/>：指定数据库用户名，这里是 LZW

<property name="password" value="123456"/>：配置登录数据库的密码，这里是 123456

Mybatis 操作数据库前需要从 SqlSessionFactory 工厂中拿一个 sqlsession 实例才能使用配置文件中定义的 sql 方法，以下是获取方法

```
10. static {
11.     try {
12.         String resource = "mybatis-config.xml";
13.         InputStream inputStream = Resources.getResourceAsStream(resource);
14.         sqlSessionFactory = (new SqlSessionFactoryBuilder()).build(inputStream);
15.     } catch (IOException var2) {
16.         var2.printStackTrace();
17.     }
18. }
```

3.3.2curd 的 mapper 接口层 sql 语句方法

UserMapper 是一个 MyBatis 的 Mapper 接口，用于定义对 tb_user 表的数据库操作。接口中的方法使用了 MyBatis 的注解来直接编写 SQL 查询，实现增删改查操作。下面是每个方法的作用解释：

（1）登录用户和密码校验

```
1.  @Select({"select * from tb_user where username = #{username} and password = #{passwor
d}"})
2.  User select(@Param("username") String var1, @Param("password") String var2);
```

使用 @Select 注解，执行 SQL 查询：根据用户名和密码在 tb_user 表中查找用户。

返回值类型是 User 对象。如果找到匹配的记录，会将记录映射到 User 实例。

@Param("username") 和 @Param("password") 注解用于绑定 SQL 中的参数 #{username} 和 #{password}。

（2）用户名修改语句

```
@Insert({"update tb_user set username = #{updateName} where username = #{username}"})
void updateName(@Param("username") String var1, @Param("updateName") String var2);
```

更新语句：根据现有的 username，将对应记录的 username 字段更新为新的 updateName 值。

使用 @Param("username") 和 @Param("updateName") 注解来绑定参数

此外，mybatis 还支持在 mapper 映射文件中自定义编写 sql 语句，可以用此方法实现一些复杂的 sql 语句编写，以下是在 xml 文件中定义的 sql 方法

（3）学生信息修改

```
<update id="update">
        update stu
        <set>
            <if test="name!=null and name!=''">
                name = #{name},
            </if>
            <if test="age!=null">
                age = #{age},
            </if>
            <if test="phone!=null and phone!=''">
                phone = #{phone},
            </if>
            <if test="email!=null and email!=''">
                email = #{email},
            </if>
            <if test="status!=null">
                status = #{status}
            </if>
        </set>
        where id = #{id}
    </update>
```

这段 XML 配置中的 <update> 标签定义了一个 MyBatis 动态 SQL 更新语句，用于更新 stu 表中的学生信息。它通过 <set> 和 <if> 条件语句动态生成更新字段，使得只有不为空的字段才会参与更新，从而避免了不必要的字段更新。

<set> 标签用于动态生成 SET 语句。它会自动处理 SQL 语句的逗号问题，即最后一个字段不会多出逗号。

<if> 标签检查各字段属性值是否不为 null 且不为空字符串。如果条件成立，生成 name = #{name}，的 SQL 片段。

3.3.3 业务层数据处理

业务层主要负责接收前端传过来的数据并进行处理，拿到需要的语句交给 mapper 执行相应的 sql 语句，同时也会对字段信息做一些验证信息，以下是一些业务层 service 处理数据的程序代码

```
22.   public PageBean<Student> selectByPageAndCondition(int currentPage, int pageSize, St
udent student) {
23.         SqlSession sqlSession = this.factory.openSession();
24.         StudentMapper mapper = (StudentMapper)sqlSession.getMapper(StudentMapper.clas
s);
25.         int begin = (currentPage - 1) * pageSize;
26.         String name = student.getName();
27.         if (name != null && !"".equals(name)) {
28.             name = "%" + name + "%";
29.             student.setName(name);
30.         }
31.         List<Student> rows = mapper.selectByPageAndCondition(begin, pageSize, studen;
32.         int totalCount = mapper.selectTotalCountByCondition(student);
33.         PageBean<Student> pageBean = new PageBean();
34.         pageBean.setRows(rows);
35.         pageBean.setTotalCount(totalCount);
36.         sqlSession.close();
37.         return pageBean;
38.     }
```

这个方法 selectByPageAndCondition 用于实现带有条件的分页查询。它根据 currentPage（当前页码）、pageSize（每页大小）和 student（查询条件）来获取符合条件的学生信息，并返回一个封装了结果的 PageBean 对象。在其中我还实现了模糊查询数据的处理，获取 student 对象中的 name 字段，用于模糊查询》如果 name 不为 null 且不为空字符串，则将 name 变为模糊查询格式（%name%），并重新设置回 student，以便在 SQL 查询中匹配包含该字符串的记录。最后创建一个 PageBean<Student> 对象 pageBean，并将查询结果 rows 设置为分页数据，同时设置总记录数 totalCount 返回给前端便于处理。

3.3.4 控制层请求处理与响应

控制层使用了 springmvc 框架，该框架封装了原生的 serlvet，所有请求都由中央 serlvet：DispatcherServlet 来处理与转发，以下是几个 controller 层获取相应请求的程序代码

（1）用户注册请求

```
@PostMapping("/register")
    public String register(
            @RequestParam("username") String username,
            @RequestParam("password") String password,
            @RequestParam("checkCode") String checkCode,
            HttpSession session) {

        String checkCodeGen = (String) session.getAttribute("checkCodeGen");
        if (!checkCodeGen.equalsIgnoreCase(checkCode)) {
            session.setAttribute("register_msg", "验证码错误");
            return "redirect:/register.jsp";
        }

        User user = new User();
        user.setUsername(username);
        user.setPassword(password);

        boolean flag = userService.register(user);
        if (flag) {
            session.setAttribute("register_msg", "注册成功,请登陆");
            return "redirect:/login.jsp";
        } else {
            session.setAttribute("register_msg", "用户名已存在,请重新注册");
            return "redirect:/register.jsp";
        }
    }
```

@PostMapping("/register")：将该方法映射到 /register 路径，并限定请求方式为 POST，处理注册请求。

@RequestParam 注解用于绑定前端表单参数到方法的形参。

username、password 和 checkCode 分别接收前端传来的用户名、密码和验证码。

HttpSession session：通过会话对象 session，可以访问和存储用户相关的数

据（例如验证码和注册反馈信息）。

创建 User 对象 user，并将前端传来的 username 和 password 设置到 user 对象中，调用 userService 的 register 方法，将用户信息保存到数据库中

如果 flag 为 true（注册成功），设置 session 属性 register_msg 为 "注册成功,请登陆"，并重定向到登录页面 /login.jsp。否则将 register_msg 设置为 "用户名已存在,请重新注册"，并重定向回注册页面 /register.jsp

（2）批量删除

```
@PostMapping("/deleteByIds")
    @ResponseBody
    public String deleteByIds(@RequestBody int[] ids) {
        studentService.deleteByIds(ids);
        return "success";
    }
```

@RequestBody 注解用于将前端传递的 JSON 数据直接转换为方法参数。

int[] ids 参数接收一个整数数组，包含要删除的记录的 ID 数组

调用 studentService 的 deleteByIds 方法，将 ids 数组传入，以执行批量删除操作。 deleteByIds 方法在 studentService 中实现，负责在数据库中删除对应 ID 的学生记录。

3.3.5 前端信息展示

登录与注册界面是使用原始的 jsp 来实现的，同时使用 jstl 来获取和存放 response 和 request 中的数据，这样可以使页面没有过多的 java 代码，方便阅读

```
<div id="loginDiv" style="height: 350px">
    <form action="user/login" id="form">
        <h1 id="loginMsg">LOGIN IN</h1>
        <div id="errorMsg">${error}${login_msg}${register_msg}</div>
        <p>用户
名:<input id="username" name="username" value="${cookie.username.value}" type="text"></p>
        <p>密
码:<input id="password" name="password" value="${cookie.password.value}" type="password"></p>
        <p>记住我:<input id="remember" name="remember" value="1" type="checkbox"></p>
        <div id="subDiv">
            <input type="submit" class="button" value="登陆">
            <input type="reset" class="button" value="取消">   
            <a href="../../register.jsp">没有账号？</a>
        </div>
```

```
    </form>
</div>
```

学生信息展示页面使用了 axios 技术来与后端交换数据，这样一来就完全出去了前端页面中的 html 代码，使前后端可以完全分离，同时使用 json 格式约定传输的数据格式，更加便于代码的维护和功能改进

```
// 查询方法
            onSubmit() {
                //console.log(this.brand);
                var _this = this;
                axios({
                    method: "post",
                    url: "student/selectByPageAndCondition?currentPage=" + _this.currentPage+"&pageSize=" + _this.pageSize,
                    data:_this.student
                }).then(function (resp) {
                   // console.log(resp.data.rows)
                    _this.tableData = resp.data.rows;
                    _this.totalCount = resp.data.totalCount;
                })
            },
```

onSubmit 方法用于提交查询请求，向后端发送包含分页参数（currentPage 和 pageSize）和查询条件（student 对象）的 POST 请求。请求成功后，将返回的数据（rows 和 totalCount）赋值给 Vue 实例的属性 tableData 和 totalCount，从而更新页面中的数据表格和分页信息。

四、实验结果及分析

4.1 部分程序功能展示

（1）新增用户信息填写

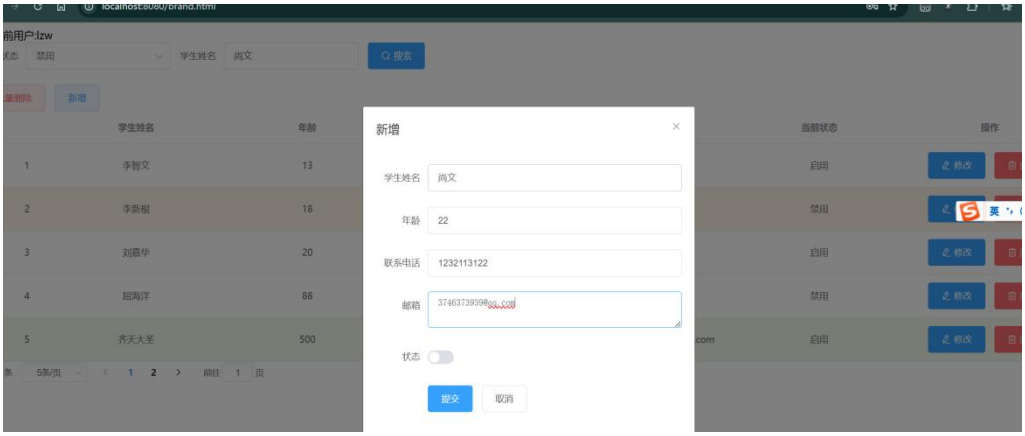

图 5：添加学生信息

图 6：新学生信息展示

（2）查询功能实现

如图查询学生中包含“李”字的学生信息

图 7：学生信息查询

（3）学生信息修改

由于使用 vue 框架，数据通过模型双向绑定，这样我们就可以做到点击修改时将现在学生的信息自动绑定到修改的表单项中，可帮助用户更准确的修改相应信息从而不用把其他不需要修改的信息再填写一遍。具体实现如下：

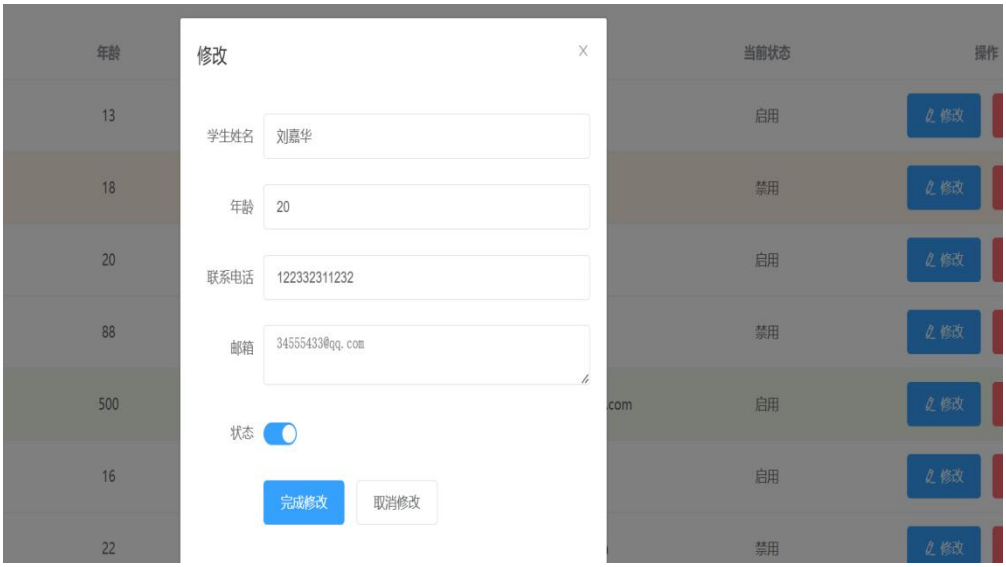

图 8：学生信息修改

4.2 项目总结

这是一个由 ssm 框架实现的一个学生信息管理系统，数据库用的是这学期学习的 oracle 数据库，在连接该数据库，操作该数据库的时候，我发现了其和 mysql 数据库操作有很多不一样的地方，例如在 mysql 中我可以直接使用 limit 来进行分页查询，得到的数据可以直接封装发送给前端，而 oracle 没有 limit 分页查询，所以我采用了这种方式

```
SELECT * FROM (
    SELECT a.*, ROWNUM rnum FROM students a
    WHERE ROWNUM <= 10
) WHERE rnum >= 1;
```

这种方式通过使用 ROWNUM 来限定查询结果的行数，并且需要嵌套查询来实现分页的效果。此外，我发现 Oracle 还可以使用更复杂的分页查询方式，如使用 ROW_NUMBER() 函数，它可以在查询时按指定的排序规则为每一行生成一个唯一的行号，从而实现更灵活的分页控制。

在我看来，Oracle 和 MySQL 在处理大数据量时的性能差异也值得注意。Oracle 对于复杂查询的优化能力通常较强，但在处理简单查询时，MySQL 可能会表现得更加轻便和快速。因此，在选择数据库时，除了考虑查询语句的写法，还要考虑具体的业务需求和系统性能的要求。

总结来说，这个学生信息管理系统通过 SSM 框架的集成，实现了学生信息的增删改查功能，同时也实现了分页查询、数据校验等功能。通过该项目的开发，我不仅掌握了 SSM 框架的基本使用，也积累了使用 Oracle 数据库进行开发的经验，为我今后在数据库管理和 web 开发方面的工作打下了坚实的基础。