

**《程序设计课程实践报告》**

题 目： 学生信息管理系统

学生姓名： **马星**

学 号： 5418122020

专业班级： 计算机科学与技术(卓越)

指导教师： 周明建

时 间： 2024/7/1

分 数：

**二0二四 年 七月一日**

**目录**

[程序设计课程报告——《学生信息管理系统》 2](#_Toc171029774)

[一、题目的要求 2](#_Toc171029775)

[二、题目的设计过程 3](#_Toc171029776)

[2.1 总体设计 3](#_Toc171029777)

[(1) 流程设计 3](#_Toc171029778)

[(2) 模块设计 4](#_Toc171029779)

[(3) 界面设计 4](#_Toc171029780)

[2.2 详细设计 6](#_Toc171029781)

[(1)界面设计 6](#_Toc171029782)

[(2) 数据库设计 7](#_Toc171029783)

[(3) 附加功能 8](#_Toc171029784)

[三、编程语言的选择 9](#_Toc171029785)

[①JavaFX 9](#_Toc171029786)

[②Mybatis+Mysql 9](#_Toc171029787)

[③Maven 9](#_Toc171029788)

[④Alibaba矢量图标库 9](#_Toc171029789)

[四、代码规范的说明 9](#_Toc171029790)

[(1) 项目结构规范 9](#_Toc171029791)

[(2) 命名规范 11](#_Toc171029792)

[(3) 依赖规范 12](#_Toc171029793)

[(4) 注释 13](#_Toc171029794)

[五、核心代码的实现 14](#_Toc171029795)

[(0) 前言 14](#_Toc171029796)

[(1) 项目启动类 App核心代码 14](#_Toc171029797)

[(2) 主界面StudentController核心代码 15](#_Toc171029798)

[① 信息展示功能 15](#_Toc171029799)

[② 按钮功能区(增删改查与信息统计) 19](#_Toc171029800)

[③ 列选择区 21](#_Toc171029801)

[④ 排序功能 21](#_Toc171029802)

[(3) 数据库交互相关代码实现 21](#_Toc171029803)

[①数据库交互逻辑 21](#_Toc171029804)

[②测试数据生成 28](#_Toc171029805)

[(4) 严格的错误信息校验 30](#_Toc171029806)

[①信息校验工具类CheckUtil 30](#_Toc171029807)

[②信息校验项与代码对照表 31](#_Toc171029808)

[(5) 提示信息弹窗 34](#_Toc171029809)

[① 操作前 34](#_Toc171029810)

[② 操作时 34](#_Toc171029811)

[③ 操作后 35](#_Toc171029812)

[④ 弹窗工具类AlertUtil 35](#_Toc171029813)

[(6) 附加功能 38](#_Toc171029814)

[① Excel文件交互功能 38](#_Toc171029815)

[② 配置文件功能 44](#_Toc171029816)

[六、问题和讨论 50](#_Toc171029817)

[(1) 重复代码 50](#_Toc171029818)

[(2) 路径问题 51](#_Toc171029819)

[(3) 异常处理 51](#_Toc171029820)

[七、总结体会 51](#_Toc171029821)

# 程序设计课程报告——《学生信息管理系统》

# 一、题目的要求

试设计一学生信息管理系统，使之能提供以下功能：

（1）系统以菜单方式工作

（2）学生信息录入功能（学生信息用文件或数据库保存）

（3）学生信息浏览功能

（4） 学生信息排序、查询功能：对信息进行排序，可以按学号，按姓名，按性别，按出生年月日，按籍贯进行查询

（5）信息统计：

(a)可以按性别，按出生年，按籍贯统计学生人数

(b)可以按班级统计学生人数

(c)可以按政治面貌统计学生人数：按班级统计学生党员人数，团员人数，以及所占比例，设计并输出统计报表。

（6）学生信息的删除与修改设计要求：界面比较美观；有一定的容错能力，比如输入的成绩不在0～100之间，就提示不合法，要求重新输入。

题目要求浓缩:

设计学生信息管理系统: 要求的学生字段有(学号,姓名,性别,出生日期,籍贯,班级,政治面貌)

(1) 系统提供学生增删改查功能

(2) 系统提供学生的排序、统计功能

(3) 系统具有容错能力, 对用户操作给出提示

# 二、题目的设计过程

## 2.1 总体设计

### (1) 流程设计

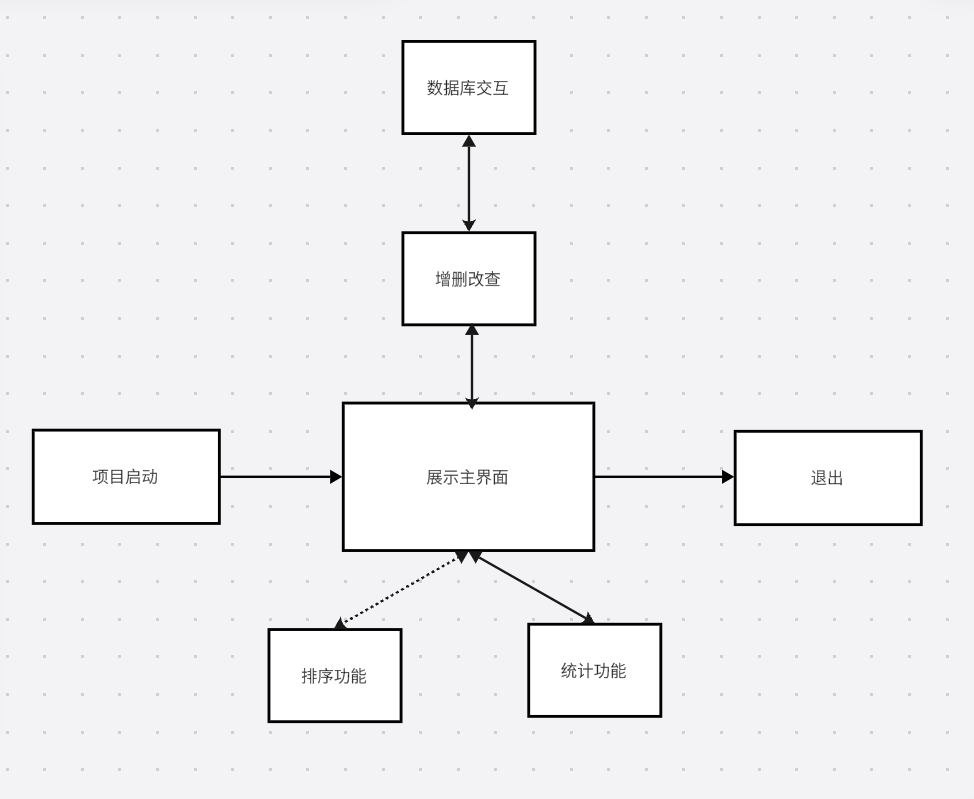
首先阅读题目要求, 确定系统需要提供的基础功能有:

(1) 学生信息的增删改查

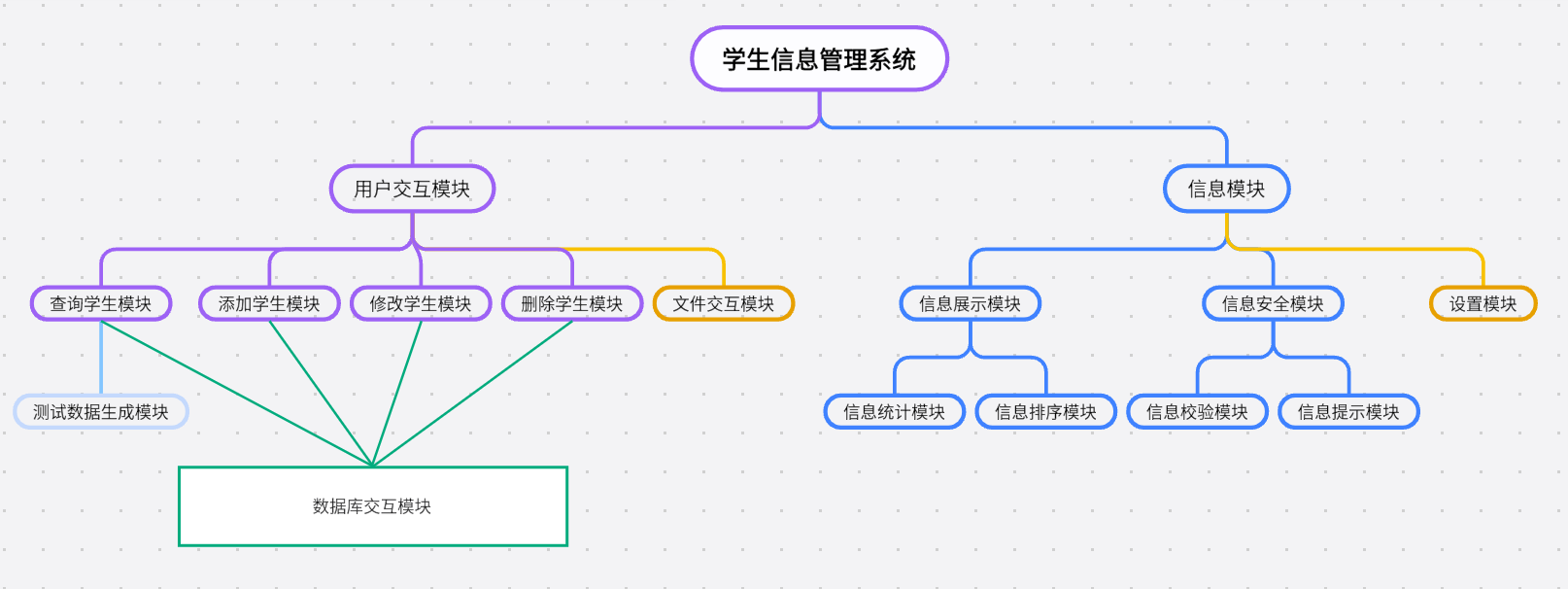
(2) 对学生信息的排序功能

(3) 对学生信息的统计功能

对此, 流程设计图如下:



### (2) 模块设计



### (3) 界面设计

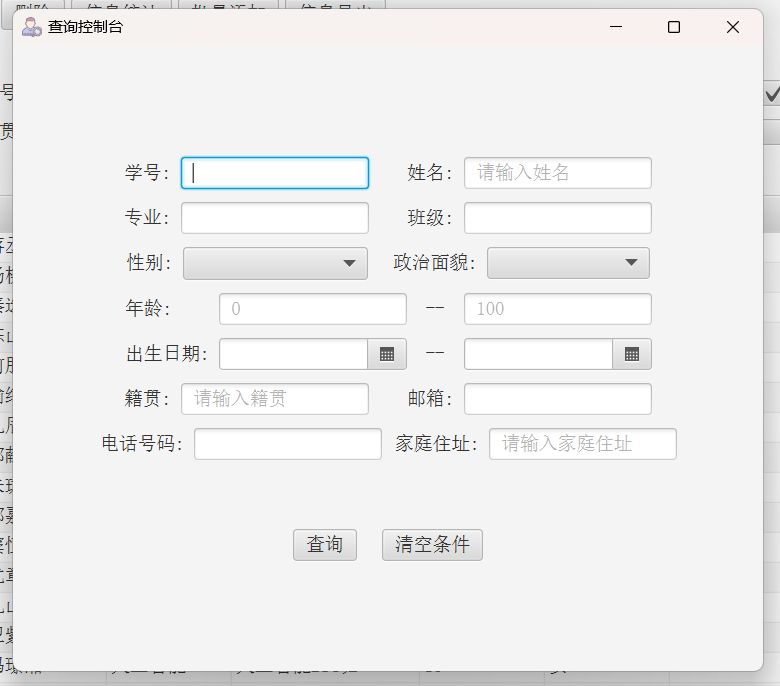
主界面分为”功能区”和”展示区”

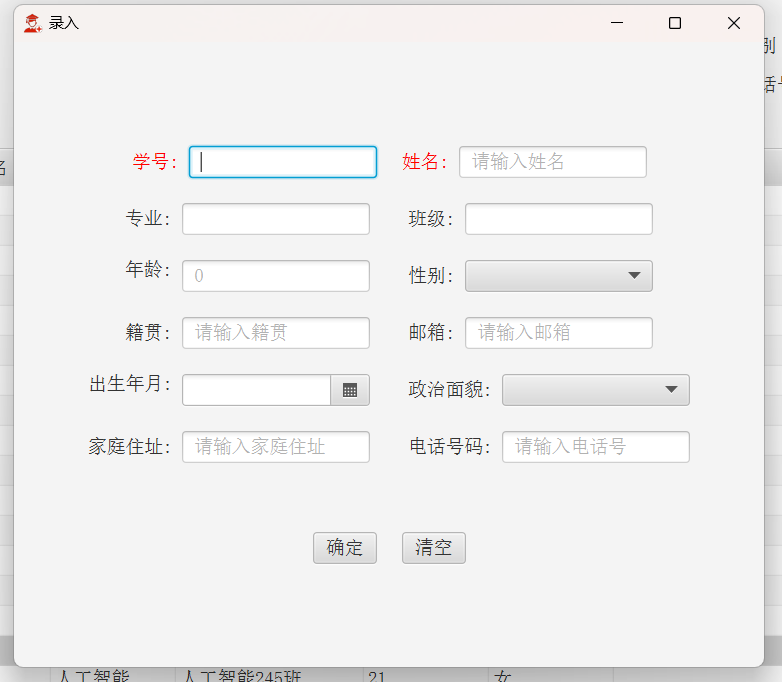
功能区: 用于实现系统的功能

展示区: 用于展示学生信息, 需要实现排序功能

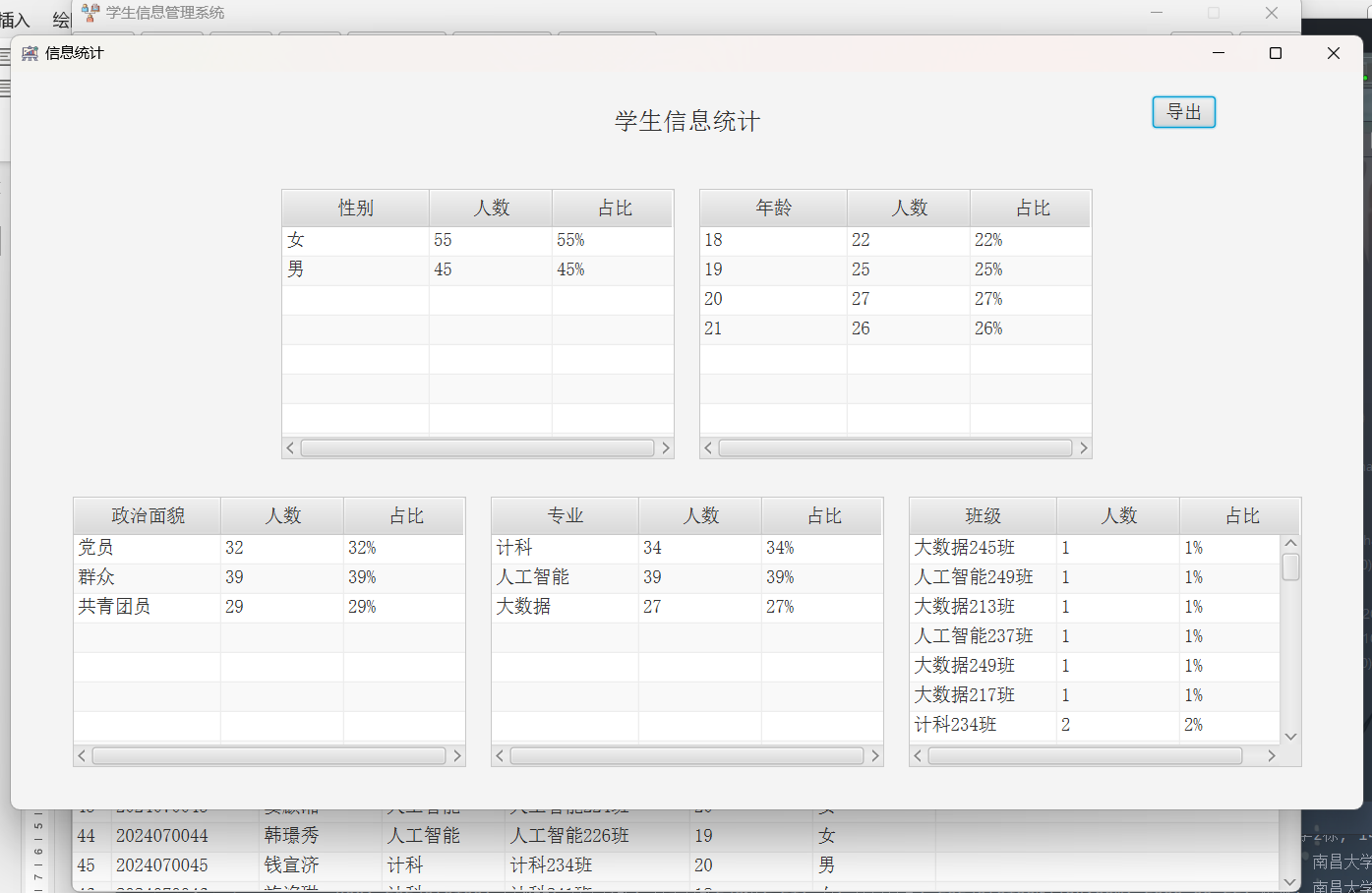


增删改查界面以表单的形式弹出, 以查询和插入为例:



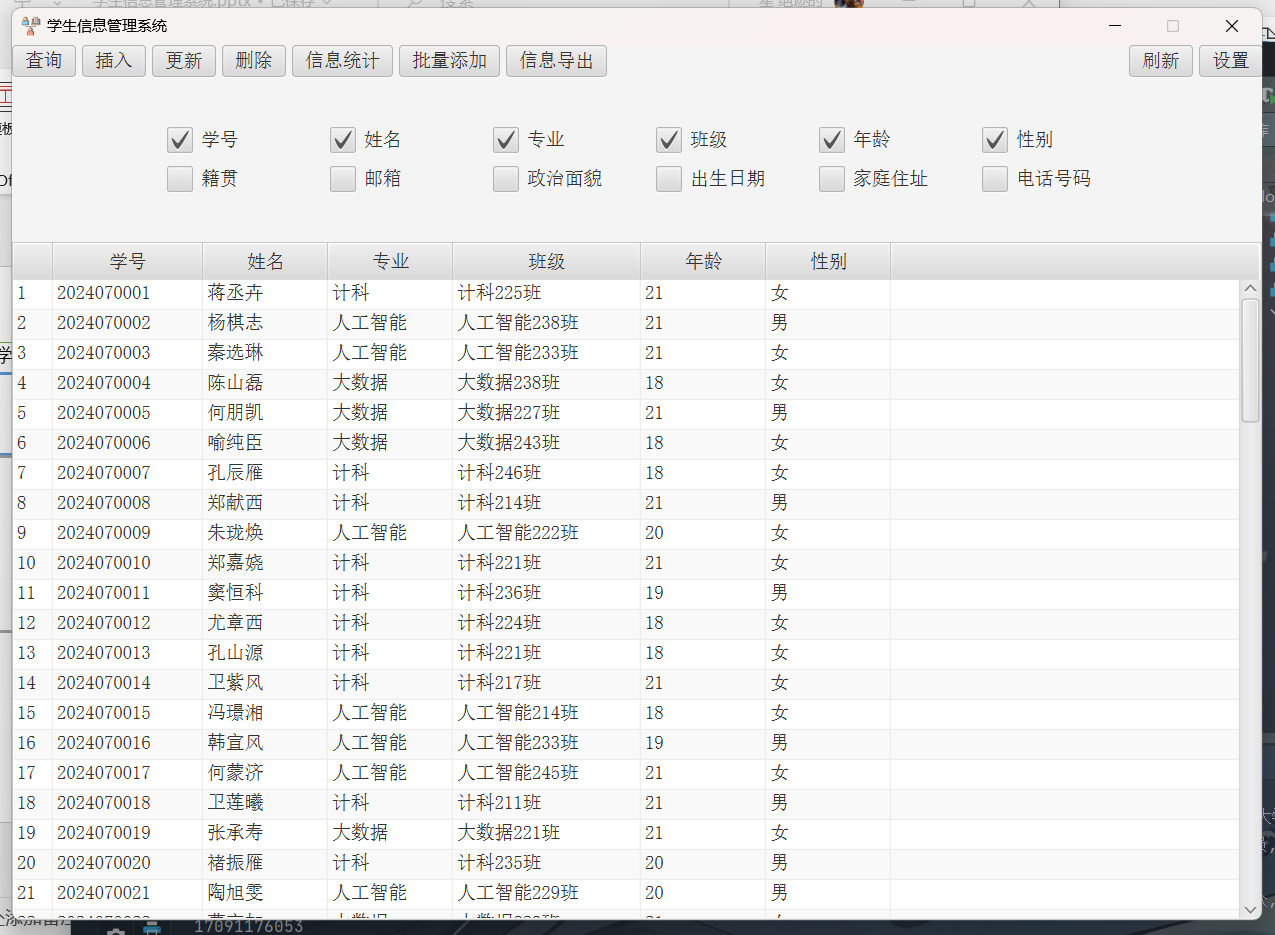


信息统计界面与主界面效果类似



## 2.2 详细设计

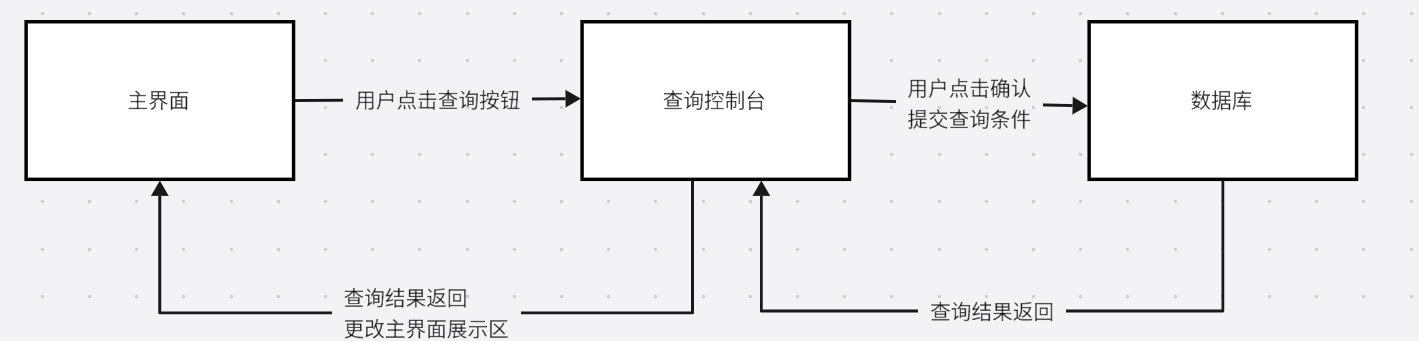
### (1)界面设计



主界面大致分三行

第一行: 功能区, 由按钮组成, 用户点击后触发相应功能

以”查询”为例, 用户点击后弹出一个查询控制台, 用户输入查询条件后进行数据库查询, 查询后数据返回到内存, 进行主界面信息展示区更新



第二行: 选择区, 由选择框组成, 与下方的信息展示区的列对应, 用户点击后修改表的显示字段

第三行: 信息展示区, 展示学生信息, 提供排序功能

### (2) 数据库设计

学生的字段有 (学号, 姓名, 专业, 班级, 年龄, 性别, 出生年月, 籍贯，邮箱, 出生日期, 政治面貌, 家庭住址，电话号码)

其中能作为候选码的只有”学号”, 所以学号为该表主码

由于系统只针对学生这一个实体, 有且仅有学号为决定因素, 规范化程度已达到4NF

字段约束条件设计如下:

学号, id, 10位数串, 非空且唯一

姓名, name, 2~4位中文字符串, 非空

专业, dept, 2~10位中文字符串

班级, className, 3位数字+"班"

年龄, age, 范围0~100

性别, sex, 男或女

籍贯, nativePlace, 2~10位中文字符串

邮箱, email, 用户名+"@"+域名

出生日期, birthday, 在2024年之前

政治面貌, outlook, 党员或共青团员或群众

家庭住址, 2~50位合法字符串

可得建表语句

USE s\_t;

DROP TABLE IF EXISTS student;

CREATE TABLE student

(

id CHAR(10) primary key comment '学号',

name VARCHAR(4) not null comment '姓名',

dept VARCHAR(10) comment '专业',

className VARCHAR(13) comment '班级',

age INT CHECK (0 <= age and age <= 100) comment '年龄',

sex CHAR(1) CHECK ( sex = '男' or sex = '女' ) comment '性别',

nativePlace VARCHAR(10) comment '籍贯',

email VARCHAR(50) comment '邮箱',

birthday DATE CHECK (birthday < now()) comment '出生日期',

outlook VARCHAR(4) CHECK ( outlook = '党员' or outlook = '共青团员' or outlook = '群众' ) comment '政治面貌',

address VARCHAR(50) comment '家庭住址',

phone CHAR(11) CHECK (phone like '1%') comment '电话号码',

CONSTRAINT chk\_class\_dept CHECK

((className is null and dept is null) or

(dept is not null and className like CONCAT(dept, '%')))

);

约束项应当在代码中先行约束, 而不是等到与数据库交互时由数据库判断

### (3) 附加功能

① 数据导入导出:

主界面功能区附加2个按钮, “批量添加” 和 “信息导出”, 用户点击后弹出文件(夹)选择框, 然后将数据 从主界面中导出 或 从excel文件导入到数据库, 提高了共享性和可移植性

依赖项: 使用第三方库org.apache.poi 进行excel文件的读取和写入

② 配置文件:

用户可在”设置”页面对项目配置进行调整, 如 弹窗的显示级别, 信息展示的列宽等

依赖项: 使用第三方库org.yaml.snakeyaml 进行yml配置文件的读取和写入

# 三、编程语言的选择

## ①JavaFX

JavaFX提供了一套全面且可扩展的用户界面组件，使得开发者能够构建具有丰富交互性的桌面应用程序, 并且提供了一个可视化的设计工具——Scene Builder，它可以简化用户界面的设计过程，使开发者能够快速构建和预览界面布局，而无需编写繁琐的代码。这提高了开发效率并降低了维护成本。

## ②Mybatis+Mysql

MyBatis作为一个持久层框架, 抽象了许多底层的数据库连接和事务管理细节，使开发者能够专注于业务逻辑。这对于学生信息管理系统这样的项目至关重要，因为这意味着可以减少出错的可能性，同时加快开发周期

## ③Maven

Maven是一个强大的项目管理和自动化构建工具，它提供了可靠的依赖管理机制。对于学生信息管理系统这样可能依赖于多个第三方库的项目，Maven可以确保所有依赖项的一致性和兼容性，避免版本冲突等问题

## ④Alibaba矢量图标库

Alibaba矢量图标库提供了大量高质量的矢量图标，这些图标可以在不同分辨率和设备上保持清晰，从而提升整体的用户界面美观度和用户体验

# 四、代码规范的说明

## (1) 项目结构规范

项目主包为src/main/java/com/jjdx/studentmange

App为项目启动类

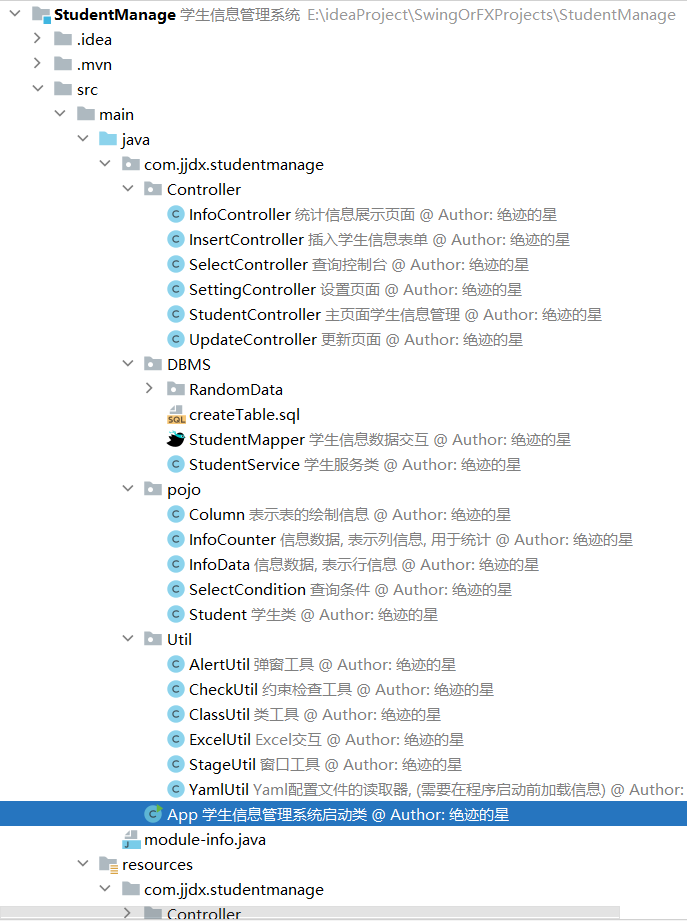
Controller包下为界面的组件控制类

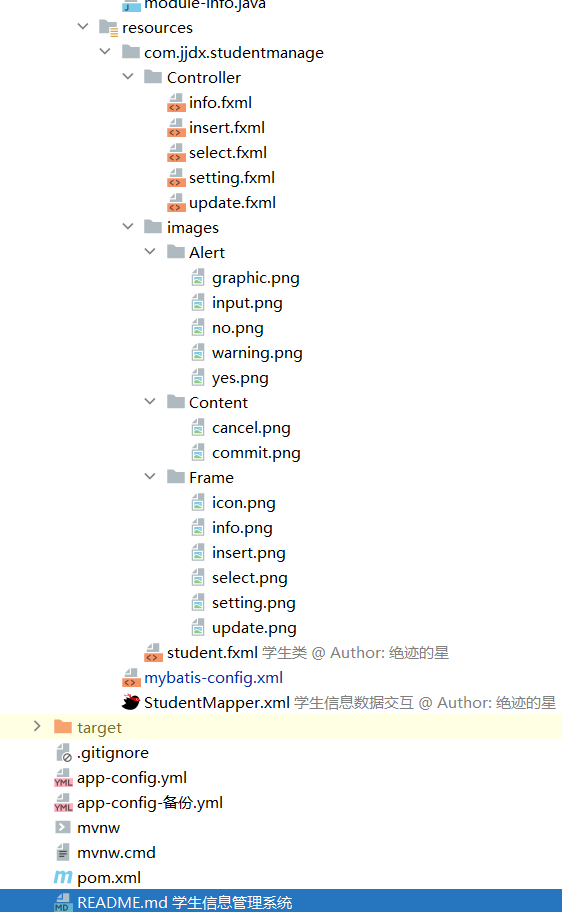
DBMS包下为数据库相关

pojo包为实体类和一些数据封装类

Util为项目的帮助工具, 通常不允许创建实例, 方法以类名.方法名的形式调用

资源类文件存放在resources目录下, 层级结构应当与主包中的包结构一致





## (2) 命名规范

类: 以大驼峰命名法命名, 例如StudentController; 除启动类外, 类都存放在相应的包中, 类名应该与包名匹配, 例如Controller包下的类均为…Controller

方法和变量: 以小驼峰命名法命名, 例如 loadInfo

返回boolean类型的方法, 命名应当为 is… 或者 not…

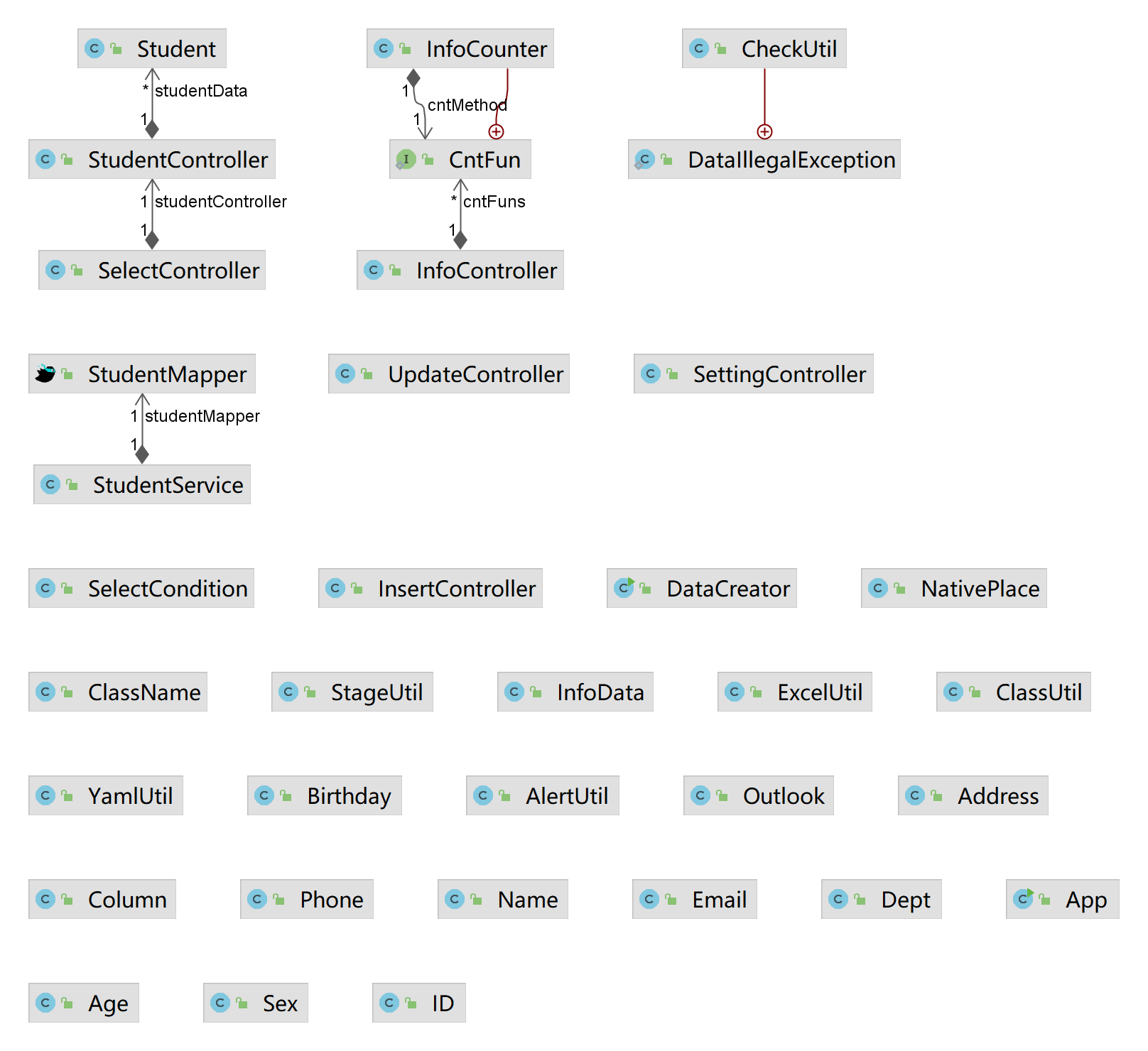
不允许使用var关键字, 类型的丢失会导致代码可读性降低, 增加代码编写与调试成本

命名应当在见名知意的情况下, 尽可能简洁明了

## (3) 依赖规范

项目应当减少依赖, 降低类与类之间的耦合度, 例如使用继承时首先考虑能否用接口的形式代替, 组合犹豫继承

类图导出如下:



## (4) 注释

代码中应当写出必要的注释, 提供从代码中无法知晓的信息, 例如流程信息等

在类上注明类的功能, 在方法上注明方法的功能和方法参数、返回值等信息

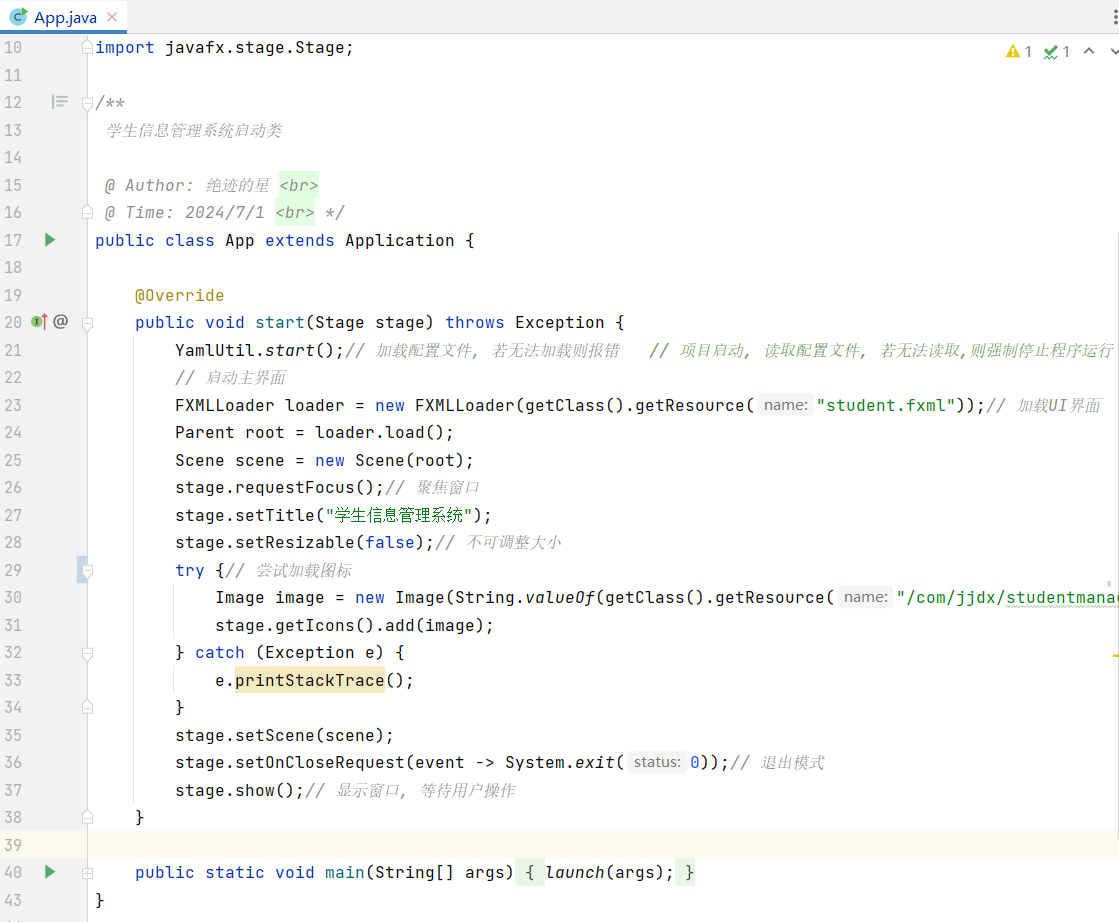
# 五、核心代码的实现

## (0) 前言

项目代码已发布在我的github仓库: <https://github.com/JueJiDeXing/StudentManage>

由于时间紧迫, 并未发布可安装的Release版本, 并且日志、动画、音效模块还未编写

## (1) 项目启动类 App核心代码



@Override

public void start(Stage stage) throws Exception {

YamlUtil.start();// 加载配置文件, 若无法加载则报错

// 启动主界面

FXMLLoader loader = new FXMLLoader(getClass().getResource("student.fxml"));// 加载UI界面

Parent root = loader.load();

root.requestFocus();// 聚焦窗口

Scene scene = new Scene(root);

stage.setTitle("学生信息管理系统");

stage.setResizable(false);// 不可调整大小

try {

Image image = new Image(String.valueOf(getClass().getResource("/com/jjdx/studentmanage/images/Frame/icon.png")));

stage.getIcons().add(image);

} catch (Exception e) {

e.printStackTrace();

}

stage.setScene(scene);

stage.setOnCloseRequest(event -> System.exit(0));// 退出模式

stage.show();// 显示窗口, 等待用户操作

}

## (2) 主界面StudentController核心代码

### ① 信息展示功能

在展示到页面上时, 先获取到 属性列数据, 包含(列字段名, 列数据源属性名, 列宽, 优先级, 是否绘制) 和 学生数据, 然后根据列数据绘制表格, 以学生数据作为数据源写入即可, 由于学生数据需要频繁使用, 所以需要实时维护当前展示的数据





/\*\*

初始化

\*/

@FXML

public void initialize() {

setData(StudentService.findAllStudent()); // 设置表格初始值

}

/\*\*

获取列数据

\*/

private List<Column> getColumnList() {

List<Column> dataList = new ArrayList<>();

List<CheckBox> checkBoxes = List.of(checkId, checkName, checkDept, checkClassName, checkAge, checkSex,

checkNativePlace, checkBirthday, checkOutlook, checkAddress, checkPhone, checkEmail);// 勾选框

HashMap<String, Object> fieldMap = YamlUtil.getLastMap("tableView.columns");// 字段名 -> 数据对

for (CheckBox checkBox : checkBoxes) {

String name = checkBox.getText();// 列字段名

HashMap<String, Object> map = (HashMap<String, Object>) fieldMap.get(name);// 获取该字段的数据对

String attribute = (String) map.get("attribute");// 对应的属性

Integer prefWidth = (Integer) map.get("prefWidth");// 列宽

Integer priority = (Integer) map.get("priority");// 优先级

boolean selected = checkBox.isSelected();// 是否绘制

dataList.add(new Column(name, attribute, prefWidth, priority, selected));

}

return dataList;

}

/\*\*

设置表格数据

@param studentData 要展示的学生数据

\*/

public void setData(List<Student> studentData) {

// 清空数据

tableView.getColumns().clear();

tableView.getItems().clear();

// 获取数据

this.studentData = studentData;//studentData维护当前展示的学生数据

List<Column> columnData = getColumnList();// 获取列数据

// 创建列并设置列名

// 序号列

TableColumn<Student, Integer> indexColumn = new TableColumn<>("");

indexColumn.setPrefWidth(80);

indexColumn.setCellValueFactory(param -> {

int value = tableView.getItems().indexOf(param.getValue()) + 1;// 序号

return new SimpleIntegerProperty(value).asObject();

});

tableView.getColumns().add(indexColumn);

// 数据列

for (Column data : columnData) {

if (!data.needDraw) continue;// 不需要绘制

TableColumn<Student, Integer> c = new TableColumn<>(data.title);// 列字段名

c.setCellValueFactory(new PropertyValueFactory<>(data.attribute));// 列属性名

c.setPrefWidth(data.prefWidth);// 列宽

tableView.getColumns().add(c);

}

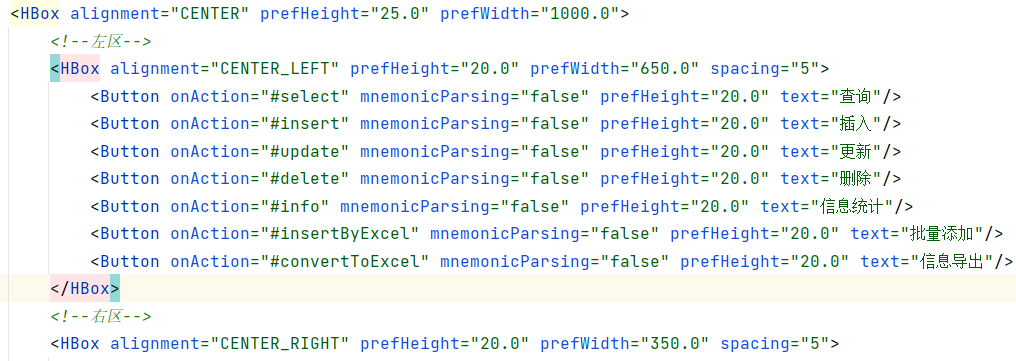
// 填充数据到表格视图

tableView.getItems().addAll(studentData);

}

### ② 按钮功能区(增删改查与信息统计)

在fxml文件中, 给Button绑定鼠标点击事件, 点击后触发新页面的开启, 以”查询”为例



首先绑定select触发事件, 用户点击后触发select函数



StageUtil.loadStage为自定义的工具方法, 具体实现如下:

****

/\*\*

弹出查询控制台, 选定查询条件

\*/

public void select(ActionEvent actionEvent) throws IOException {

FXMLLoader loader = StageUtil.loadStage(

getClass(), "select.fxml", selectStage,

"查询控制台", "/com/jjdx/studentmanage/images/Frame/select.png");

//设置回调, select出的数据需要回到主界面

if (loader != null) {

((SelectController) loader.getController()).setDataCallback(this);

}

}

/\*\*

加载页面, 如果已加载则直接返回null

@param clazz 当前类

@param xmlPath 页面路径

@param stage 当前窗口

@param title 窗口标题

@param iconPath 图标路径

\*/

public static FXMLLoader loadStage(Class clazz, String xmlPath, Stage stage, String title, String iconPath) throws IOException {

if (stage.isShowing()) {// 已在展示, 防止多开

stage.requestFocus();

return null;

}

try {// 尝试加载图标

stage.getIcons().add(new Image(String.valueOf(clazz.getResource(iconPath))));

} catch (Exception e) {

e.printStackTrace();

}

// 加载UI

FXMLLoader loader = new FXMLLoader(clazz.getResource(xmlPath));

Parent root = loader.load();

Scene scene = new Scene(root);

stage.setScene(scene);

stage.setTitle(title);

stage.show();

return loader;// 将loader返回出去

}

### ③ 列选择区

由于类内实时维护了Student数据, 所以只需要重新获取属性列信息, 再将数据绘制到TableView中

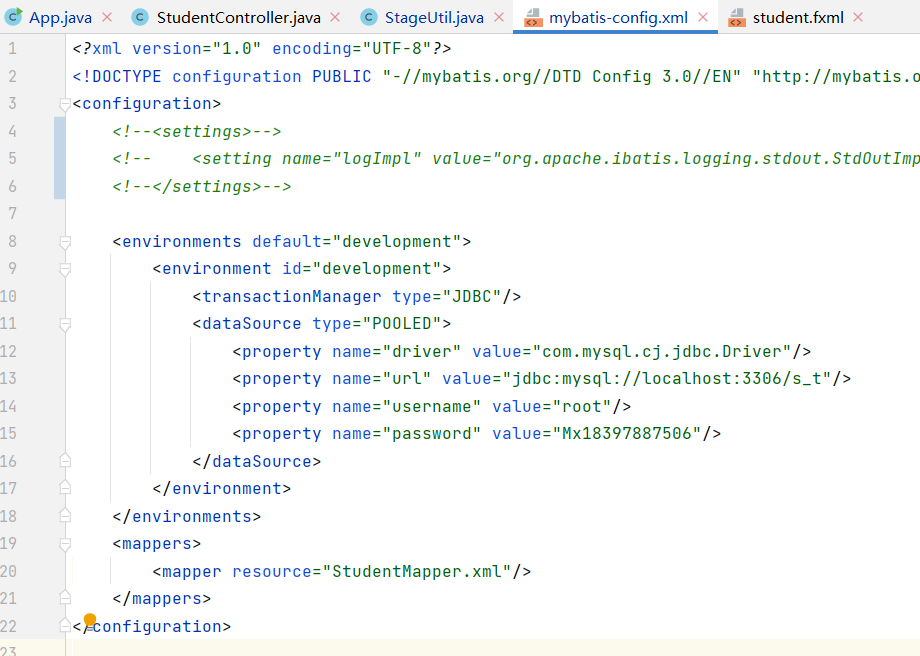
### ④ 排序功能

TableView组件自带排序功能, 不需要编写

## (3) 数据库交互相关代码实现

### ①数据库交互逻辑

1. 编写mybatis的配置文件



<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN" "http://mybatis.org/dtd/mybatis-3-config.dtd">

<configuration>

<environments default="development">

<environment id="development">

<transactionManager type="JDBC"/>

<dataSource type="POOLED">

<property name="driver" value="com.mysql.cj.jdbc.Driver"/>

<property name="url" value="jdbc:mysql://localhost:3306/s\_t"/>

<property name="username" value="\*\*\*"/>

<property name="password" value="\*\*\*"/>

</dataSource>

</environment>

</environments>

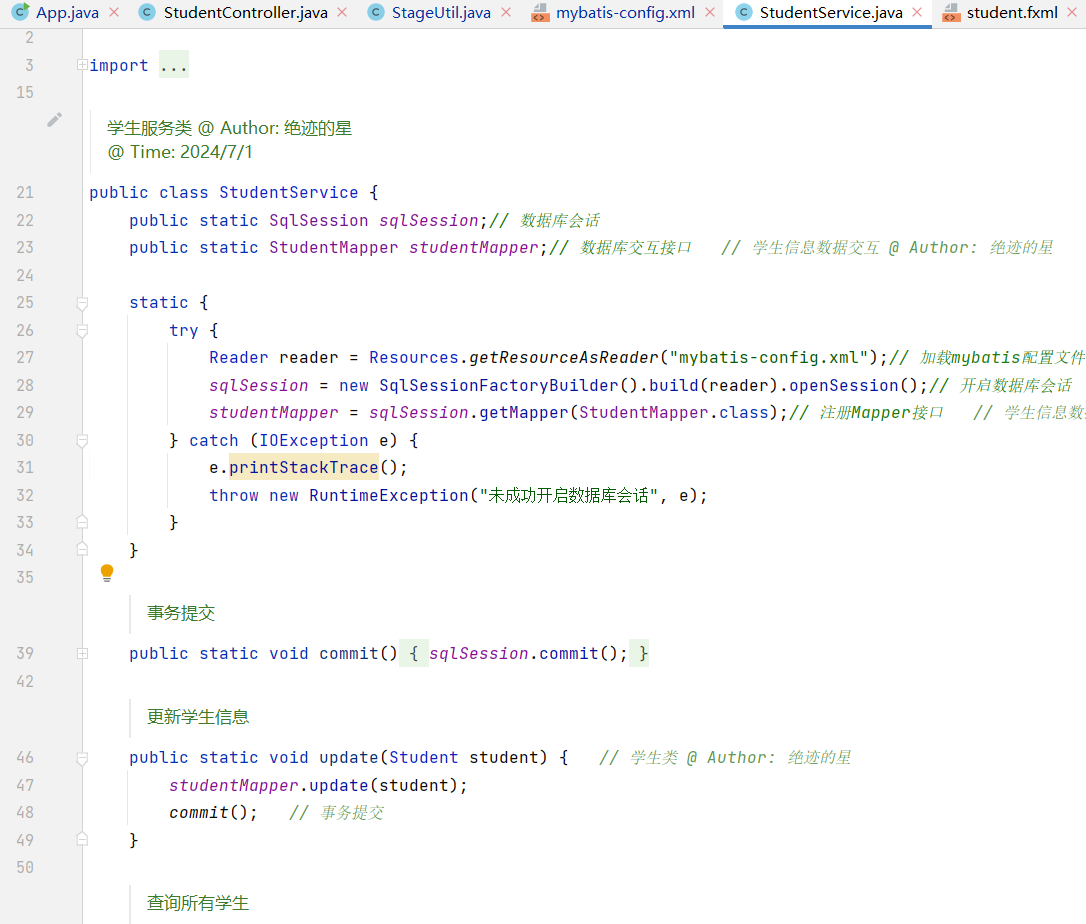
<mappers>

<mapper resource="StudentMapper.xml"/>

</mappers>

</configuration>

2. 使用SqlSessionFactoryBuilder构建SqlSessionFactory,开启数据库会话SqlSession, 并注册一个数据库交互类StudentMapper



package com.jjdx.studentmanage.DBMS;

import com.jjdx.studentmanage.Util.CheckUtil;

import com.jjdx.studentmanage.pojo.SelectCondition;

import com.jjdx.studentmanage.pojo.Student;

import org.apache.ibatis.io.Resources;

import org.apache.ibatis.session.SqlSession;

import org.apache.ibatis.session.SqlSessionFactoryBuilder;

import java.io.IOException;

import java.io.Reader;

import java.util.HashMap;

import java.util.HashSet;

import java.util.List;

/\*\*

学生服务类

@ Author: 绝迹的星 <br>

@ Time: 2024/7/1 <br> \*/

public class StudentService {

public static SqlSession sqlSession;// 数据库会话

public static StudentMapper studentMapper;// 数据库交互接口

static {

try {

Reader reader = Resources.getResourceAsReader("mybatis-config.xml");// 加载mybatis配置文件

sqlSession = new SqlSessionFactoryBuilder().build(reader).openSession();// 开启数据库会话

studentMapper = sqlSession.getMapper(StudentMapper.class);// 注册Mapper接口

} catch (IOException e) {

e.printStackTrace();

throw new RuntimeException("未成功开启数据库会话", e);

}

}

/\*\*

事务提交

\*/

public static void commit() {

sqlSession.commit();

}

/\*\*

更新学生信息

\*/

public static void update(Student student) {

studentMapper.update(student);

commit();

}

/\*\*

查询所有学生

\*/

public static List<Student> findAllStudent() {

return studentMapper.findAllStudent();

}

/\*\*

查询学生

\*/

public static List<Student> queryStudents(SelectCondition condition) throws CheckUtil.DataIllegalException {

String info = CheckUtil.conditionValidInfo(condition);

if (info != null) {

throw new CheckUtil.DataIllegalException(info);

}

return studentMapper.queryStudents(condition);

}

/\*\*

插入学生

@param student 学生信息

@return 插入结果, null为插入成功

\*/

public static String insert(Student student) {

String valid = CheckUtil.studentInsertValid(student);

if (valid != null) return valid;

try {

studentMapper.insert(student);

commit();

return null;

} catch (Exception e) {

return "插入失败";

}

}

/\*\*

插入学生

@param studentList 学生信息列表

@return 插入结果, 以哈希表形式返回, [failCnt = int, failInfo = HashSet< String >], 如果返回值为null,则表示全部插入成功

\*/

public static HashMap<String, Object> insert(List<Student> studentList) {

HashMap<String, Object> info = new HashMap<>();

info.put("failCnt", 0);

info.put("failInfo", new HashSet<>());

for (Student student : studentList) {

String result = insert(student);

if (result == null) continue;

// 插入失败

info.put("failCnt", (int) info.get("failCnt") + 1);// 失败计数

HashSet<String> set = (HashSet<String>) info.get("failInfo");// 失败信息

set.add(result);

}

if ((int) info.get("failCnt") == 0) return null;// 全部成功

return info;

}

/\*\*

删除学生信息

\*/

public static void delete(String id) {

studentMapper.delete(id);

commit();

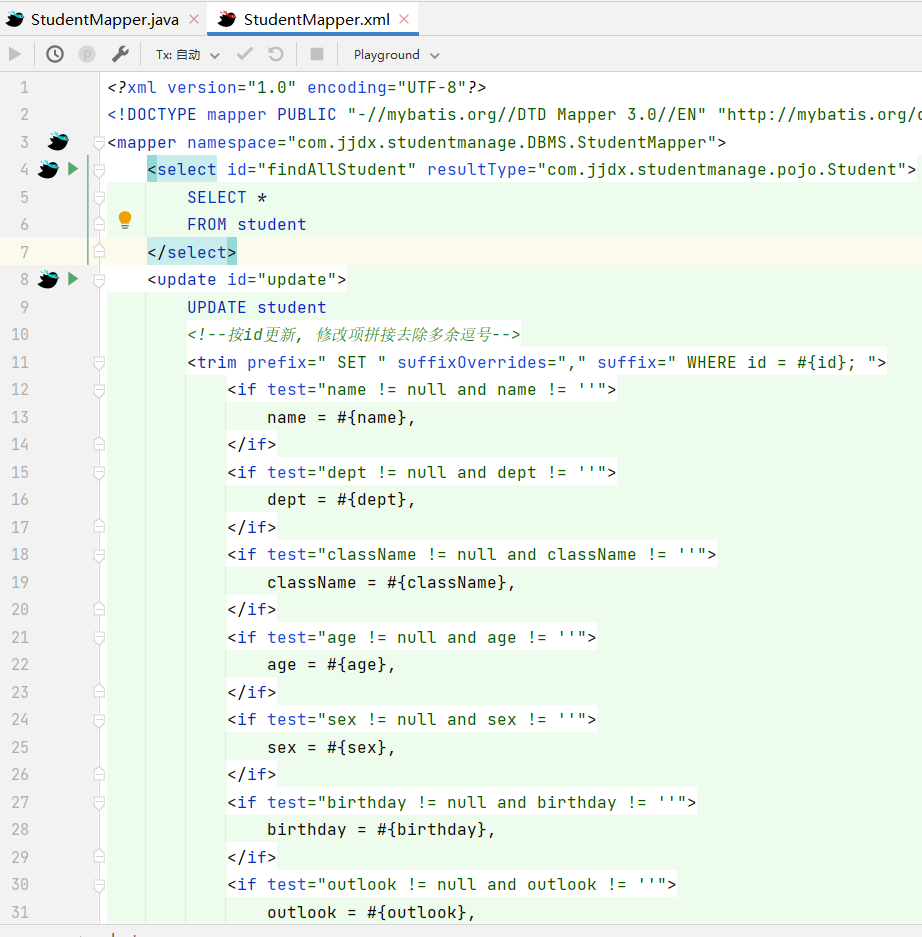
}

}

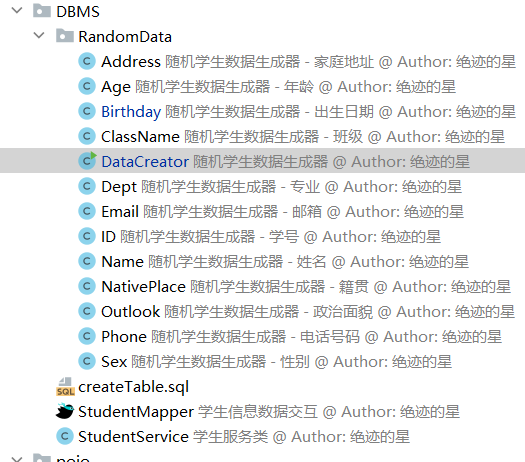
3. 在StudentMapper类中编写接口, 然后对外提供数据库交互方法



4. 在对应的xml文件中编写实际的sql语句

****

### ②测试数据生成



在DBMS模块的RandomData下, 定义DataCreator类, 运行其main方法, 通过create方法生成学生数据, 再通过StudentService的数据库交互接口insert到数据库中

package com.jjdx.studentmanage.DBMS.RandomData;

import com.jjdx.studentmanage.pojo.Student;

import java.time.LocalDate;

import java.util.ArrayList;

import java.util.List;

public class DataCreator {

public static void main(String[] args) {

//生成一些随机学生信息

//List<Student> studentList = create(1000);

//HashMap<String, Object> insert = StudentService.insert(studentList);

//System.out.println(insert);

}

private static List<Student> create(int n) {

List<Student> studentList = new ArrayList<>();

for (int i = 0; i < n; i++) {

String id = ID.get();

String name = Name.get();

String dept = Dept.get();

String className = ClassName.get(dept);

Integer age = Age.get();

String sex = Sex.get();

String nativePlace = NativePlace.get();

String email = Email.get(id);

LocalDate birthday = Birthday.get(age);

String outlook = Outlook.get();

String address = Address.get();

String phone = Phone.get();

Student student = new Student(id, name, dept, className, age, sex, nativePlace, email, birthday, outlook, address, phone);

studentList.add(student);

}

return studentList;

}

}

其他类, 诸如ID、Name, 与学生字段一一对应, 并附带get方法返回随机合法数据

以ID类为例:

package com.jjdx.studentmanage.DBMS.RandomData;

public class ID {

static int start = 2024070001;// 初始id

public static String get() {

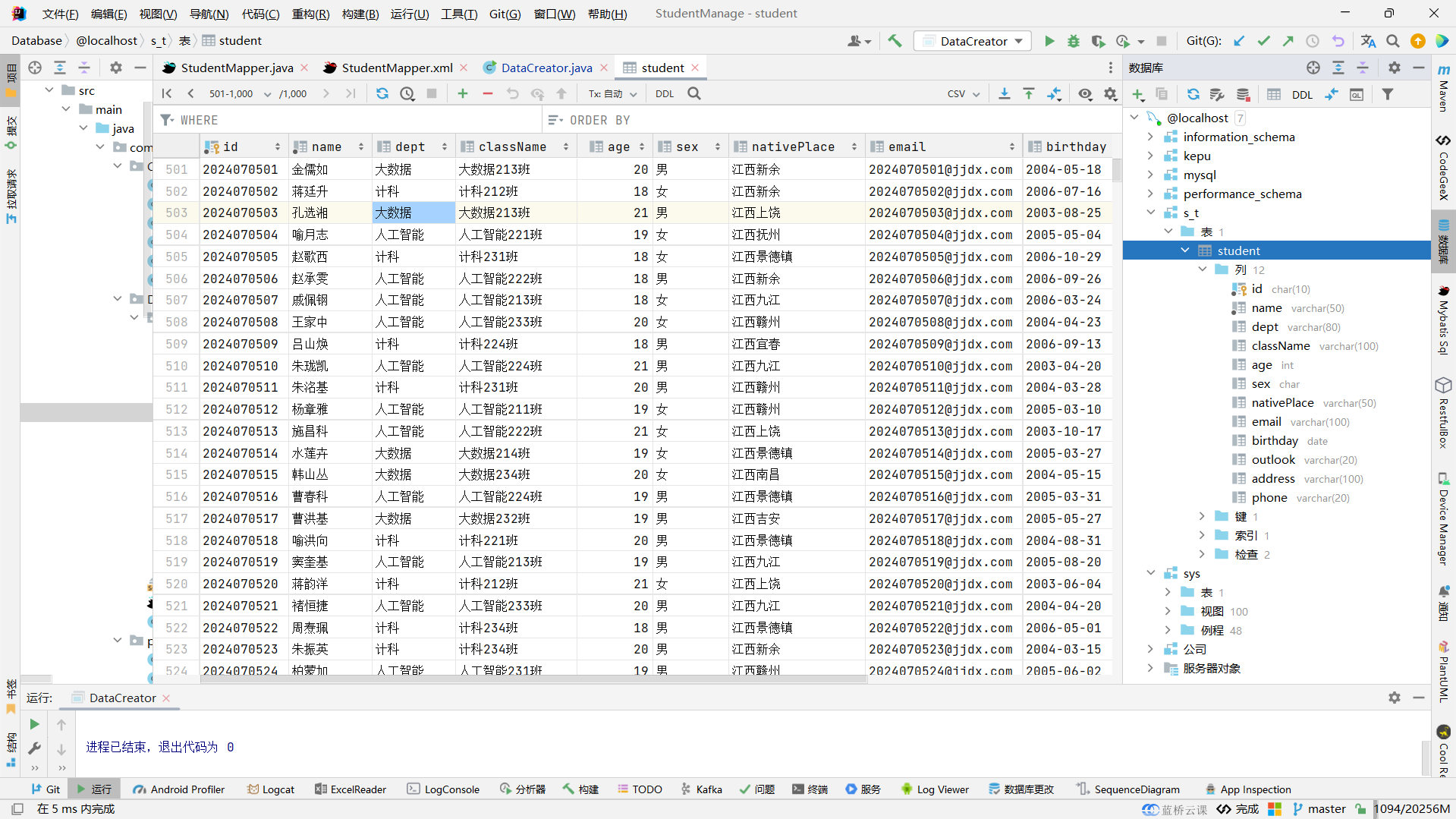
return String.valueOf(start++);

}

}

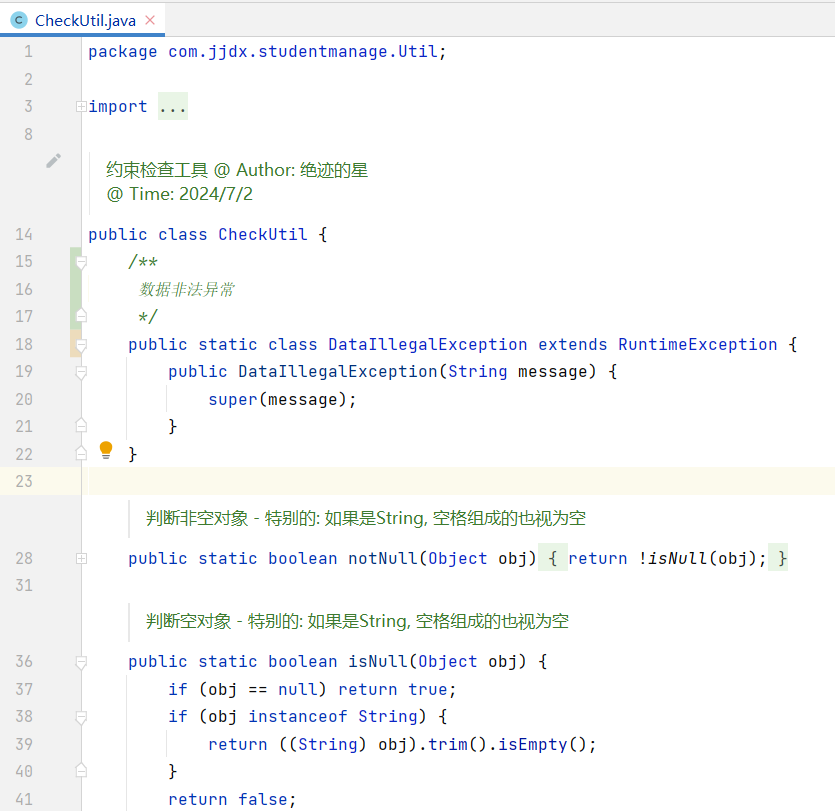
数据生成效果:

生成并插入1000条学生数据, 全部成功



## (4) 严格的错误信息校验

### ①信息校验工具类CheckUtil





### ②信息校验项与代码对照表

|  |  |
| --- | --- |
| 字段 | CheckUtil相关代码 |
| 学号id | /\*\*  合法id, 10位数字  \*/  public static boolean isValidId(String id) {  return CheckUtil.notNull(id) && id.matches("\\d{10}");  } |
| 姓名name | /\*\*  合法姓名, [2,4]位数字  \*/  public static boolean isValidName(String name) {  return CheckUtil.notNull(name) && name.matches("^[\\u4e00-\\u9fa5]{2,4}$");  } |
| 专业dept | /\*\*  合法专业, [2,10]位中文字符  \*/  public static boolean isValidDept(String dept) {  return CheckUtil.notNull(dept) && dept.matches("^[\\u4e00-\\u9fa5]{2,10}$");  } |
| 班级className | /\*\*  合法班级, 专业 + 3位数字 + '班'  \*/  public static boolean isValidClassName(String className) {  return CheckUtil.notNull(className) && className.matches("^ [\\u4e00-\\u9fa5]+\\d{3}班$");  } |
| 年龄age | /\*\*  合法年龄, 0~100  \*/  public static boolean isValidAge(Integer age) {  return CheckUtil.notNull(age) && 0 <= age && age <= 100;  } |
| 性别sex | /\*\*  合法性别, 男|女  \*/  public static boolean isValidSex(String sex) {  return CheckUtil.notNull(sex) && (sex.equals("男") || sex.equals("女"));  } |
| 籍贯nativePlace | /\*\*  合法籍贯, [2,10]位中文字符  \*/  public static boolean isValidNativePlace(String nativePlace) {  return CheckUtil.notNull(nativePlace) &&  nativePlace.matches("^[\\u4e00-\\u9fa5]{2,10}$");  } |
| 邮箱email | /\*\*  合法邮箱, 用户名 + '@' + 域名  \*/  public static boolean isValidEmail(String email) {  return CheckUtil.notNull(email) &&  email.matches("^[a-zA-Z0-9\_-]+@[a-zA-Z0-9\_-]+(\\.[a-zA-Z0-9\_-]+)+$");  } |
| 出生日期birthday | /\*\*  合法出生日期, 在现在之前  \*/  public static boolean isValidBirthday(LocalDate birthday) {  return CheckUtil.notNull(birthday) &&  birthday.isBefore(LocalDate.now());  } |
| 政治面貌outlook | /\*\*  合法政治面貌, 党员|共青团员|群众  \*/  public static boolean isValidOutlook(String outlook) {  return CheckUtil.notNull(outlook) &&  (outlook.equals("党员") || outlook.equals("共青团员") || outlook.equals("群众"));  } |
| 家庭住址address | /\*\*  合法家庭住址, [2,50]位合法字符  \*/  public static boolean isValidAddress(String address) {  return CheckUtil.notNull(address) &&  address.matches("^[\\u4e00-\\u9fa5a-zA-Z0-9]{2,50}$");  } |
| 手机号phone | /\*\*  合法手机号, 以1开始的11位数字  \*/  public static boolean isValidPhone(String phone) {  return CheckUtil.notNull(phone) &&  phone.matches("^1\\d{10}$");  } |
| 学号存在性判断 | /\*\*  数据库中是否存在该id  \*/  public static boolean isExistId(String id) {  if (!isValidId(id)) return false;  for (Student student : StudentService.findAllStudent()) {  if (student.getId().equals(id)) return true;  }  return false;  } |
| 专业与班级对应 | if (notNull(s.getClassName())) {  if (!isValidClassName(s.getClassName())) return "班级不合法";  if (isNull(s.getDept())) return "需填写专业";  if (!s.getClassName().startsWith(s.getDept())) return "专业不匹配";  } |
| 年龄与出生日期对应 | if (notNull(s.getBirthday())) {  if (!isValidBirthday(s.getBirthday())) return "出生日期不合法";  if (notNull(s.getAge()) && s.getBirthday().plusYears(s.getAge()).getYear() != LocalDate.now().getYear()) {  return "年龄与出生日期不匹配";  }  } |

## (5) 提示信息弹窗

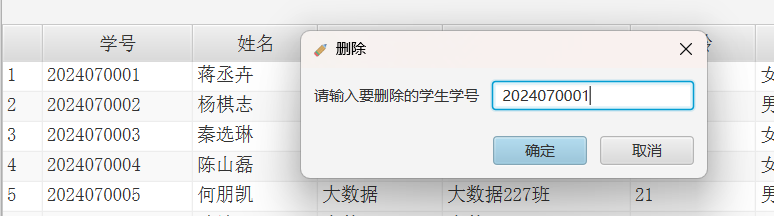
### ① 操作前

用户填写表单时, 鼠标悬浮在表单填写区会显示字段的填写要求, 以”插入”为例



### ② 操作时

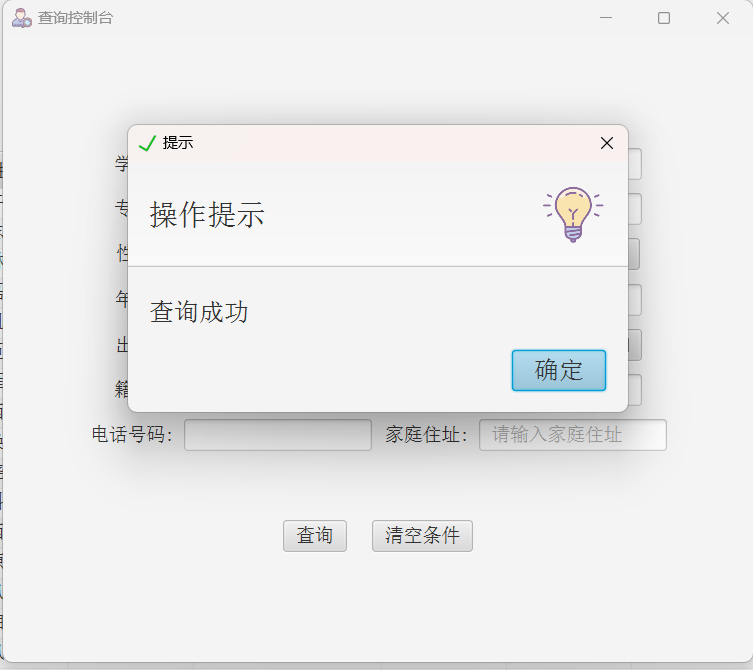
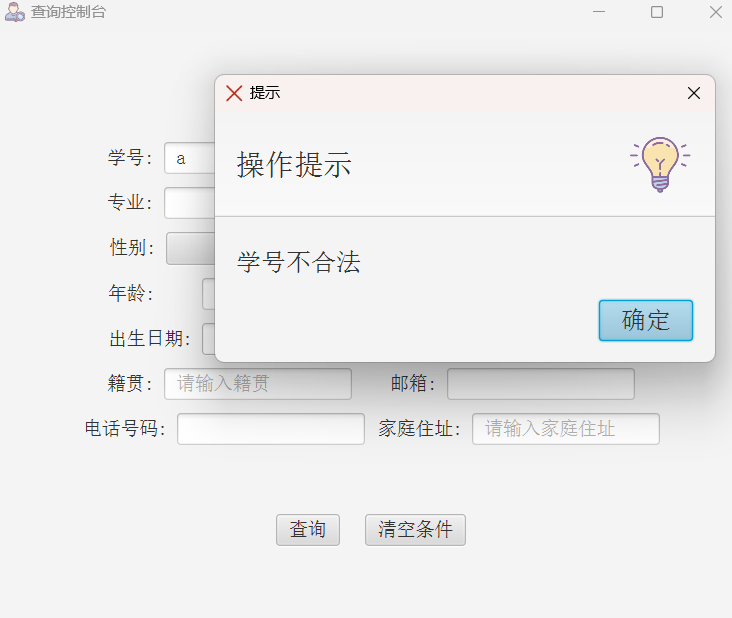
当用户执行某些影响重大的操作前会显示确认框, 如”删除”操作





### ③ 操作后

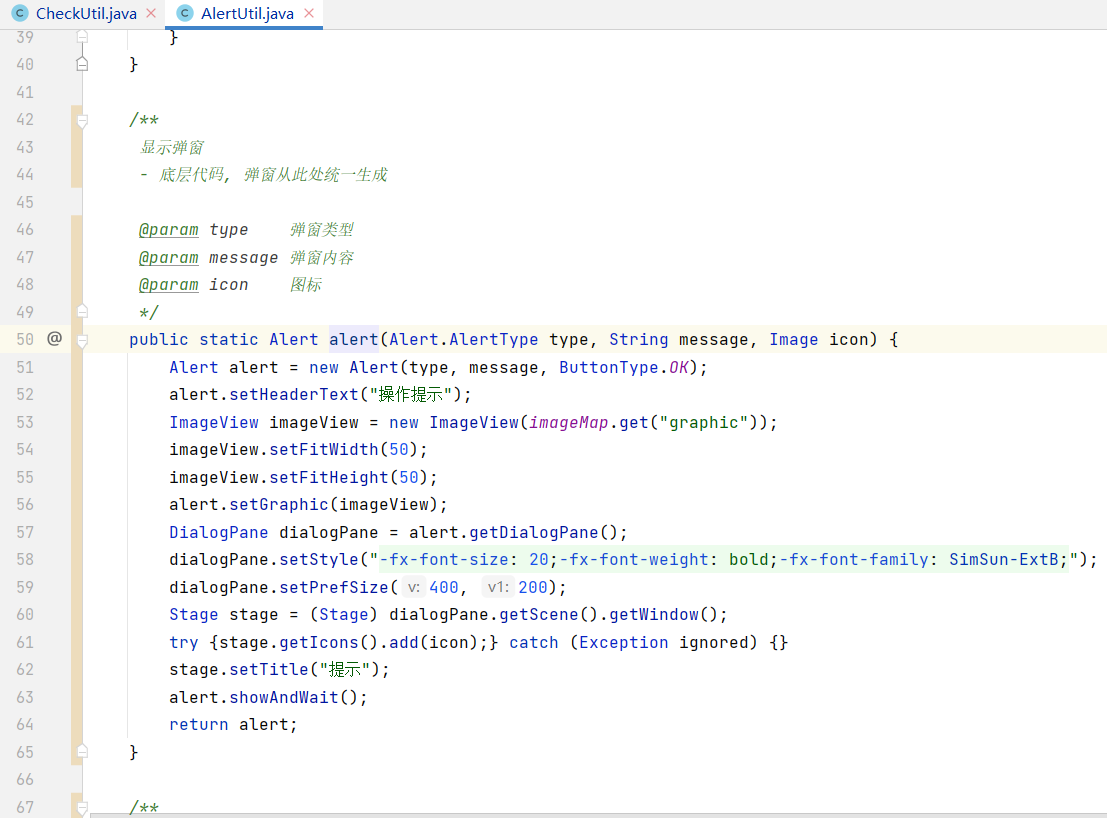
当用户执行操作后会给出提示信息弹窗

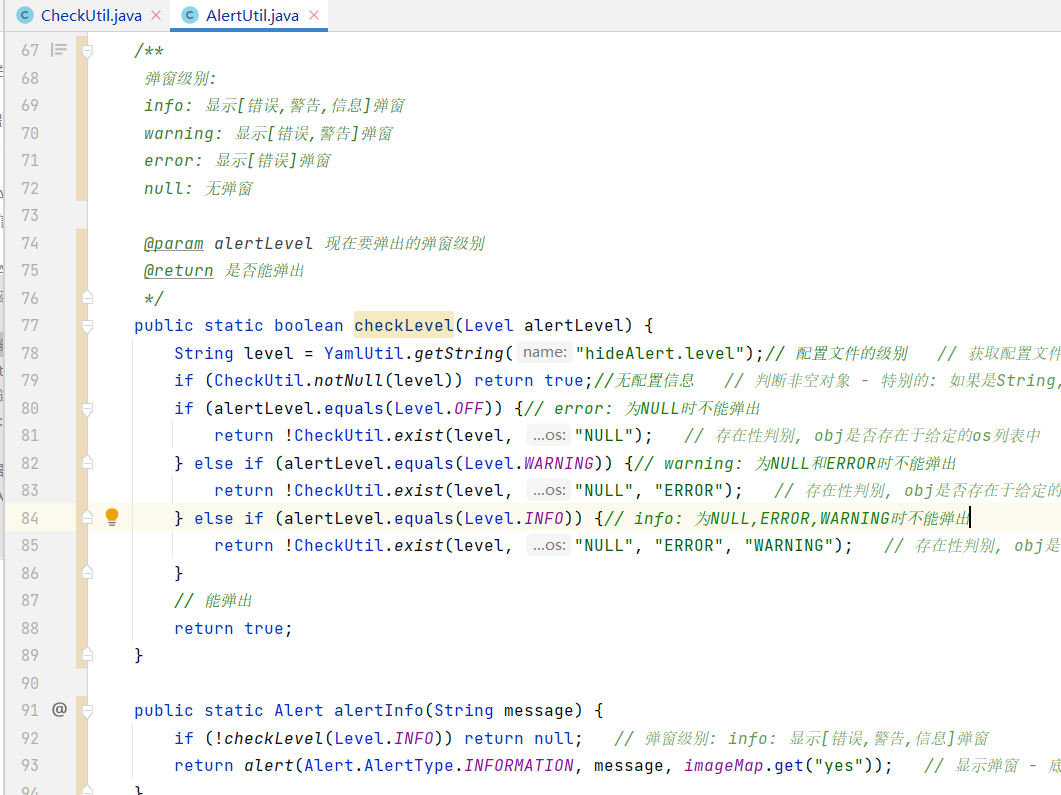
 

### ④ 弹窗工具类AlertUtil

由于弹窗使用非常频繁, 所以集成为AlertUtil工具类, 类似于工厂, 在AlertUtil中统一创建Alert弹窗, 并统一样式, 并且赋予弹窗级别, 用户可以在设置中进行更改弹窗显示级别

弹窗弹出项有校验不合格、 操作前提示、操作结果提示等情况





/\*\*

显示弹窗

- 底层代码, 弹窗从此处统一生成

@param type 弹窗类型

@param message 弹窗内容

@param icon 图标

\*/

public static Alert alert(Alert.AlertType type, String message, Image icon) {

Alert alert = new Alert(type, message, ButtonType.OK);

alert.setHeaderText("操作提示");

ImageView imageView = new ImageView(imageMap.get("graphic"));

imageView.setFitWidth(50);

imageView.setFitHeight(50);

alert.setGraphic(imageView);

DialogPane dialogPane = alert.getDialogPane();

dialogPane.setStyle("-fx-font-size: 20;-fx-font-weight: bold;-fx-font-family: SimSun-ExtB;");

dialogPane.setPrefSize(400, 200);

Stage stage = (Stage) dialogPane.getScene().getWindow();

try {stage.getIcons().add(icon);} catch (Exception ignored) {}

stage.setTitle("提示");

alert.showAndWait();

return alert;

}

/\*\*

弹窗级别:

info: 显示[错误,警告,信息]弹窗

warning: 显示[错误,警告]弹窗

error: 显示[错误]弹窗

null: 无弹窗

@param alertLevel 现在要弹出的弹窗级别

@return 是否能弹出

\*/

public static boolean checkLevel(Level alertLevel) {

String level = YamlUtil.getString("hideAlert.level");// 配置文件的级别

if (CheckUtil.notNull(level)) return true;//无配置信息

if (alertLevel.equals(Level.OFF)) {// error: 为NULL时不能弹出

return !CheckUtil.exist(level, "NULL");

} else if (alertLevel.equals(Level.WARNING)) {// warning: 为NULL和ERROR时不能弹出

return !CheckUtil.exist(level, "NULL", "ERROR");

} else if (alertLevel.equals(Level.INFO)) {// info: 为NULL,ERROR,WARNING时不能弹出

return !CheckUtil.exist(level, "NULL", "ERROR", "WARNING");

}

// 能弹出

return true;

}

public static Alert alertInfo(String message) {

if (!checkLevel(Level.INFO)) return null;

return alert(Alert.AlertType.INFORMATION, message, imageMap.get("yes"));

}

public static Alert alertWarning(String message) {

if (!checkLevel(Level.WARNING)) return null;

return alert(Alert.AlertType.WARNING, message, imageMap.get("warning"));

}

public static Alert alertError(String message) {

if (!checkLevel(Level.OFF)) return null;// 由于没有ERROR项, 拿OFF顶一下

return alert(Alert.AlertType.ERROR, message, imageMap.get("no"));

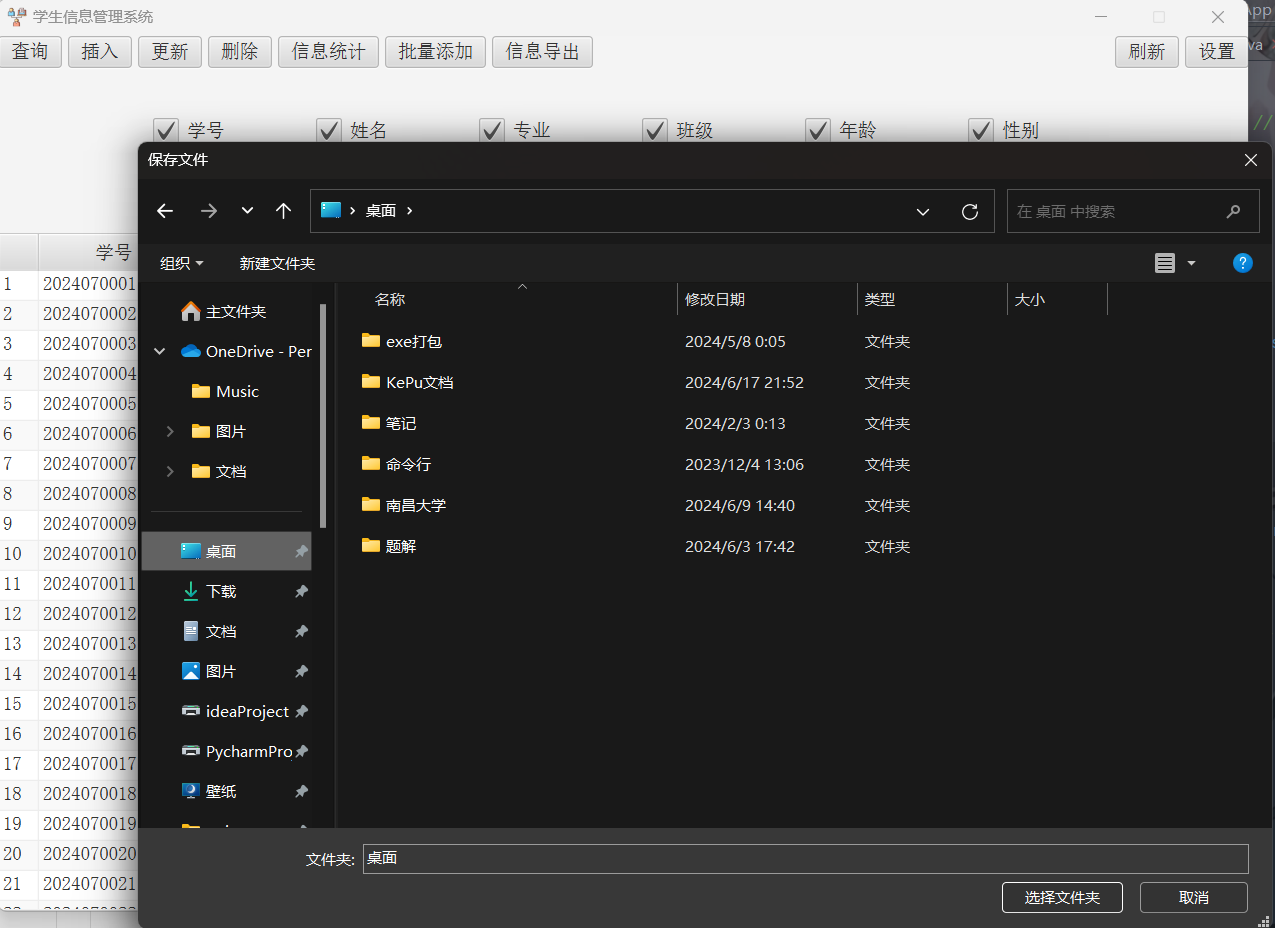
}

## (6) 附加功能

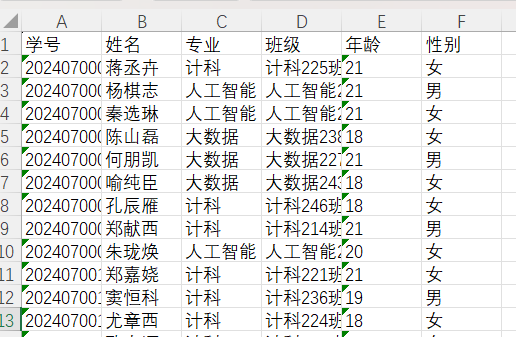
### ① Excel文件交互功能

功能展示:

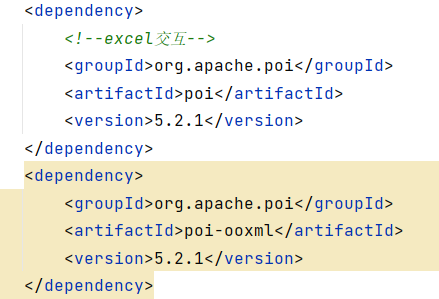
信息导出 – 选择文件保存位置 – 导出为excel文件



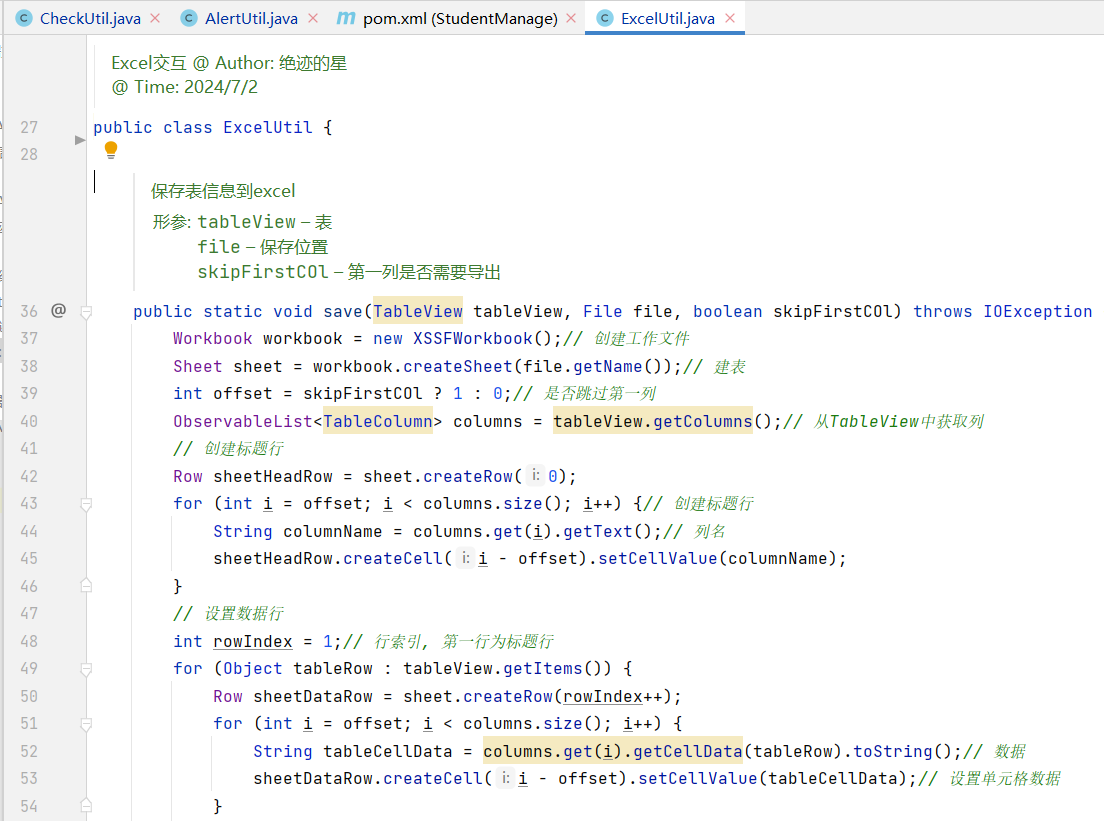




依赖项org.apache.poi和poi-ooxml的maven坐标



代码部分:



/\*\*

Excel交互

@ Author: 绝迹的星 <br>

@ Time: 2024/7/2 <br> \*/

public class ExcelUtil {

/\*\*

保存表信息到excel

@param tableView 表

@param file 保存位置

@param skipFirstCOl 第一列是否需要导出

\*/

public static void save(TableView tableView, File file, boolean skipFirstCOl) throws IOException {

Workbook workbook = new XSSFWorkbook();// 创建工作文件

Sheet sheet = workbook.createSheet(file.getName());// 建表

int offset = skipFirstCOl ? 1 : 0;// 是否跳过第一列

ObservableList<TableColumn> columns = tableView.getColumns();// 从TableView中获取列

// 创建标题行

Row sheetHeadRow = sheet.createRow(0);

for (int i = offset; i < columns.size(); i++) {// 创建标题行

String columnName = columns.get(i).getText();// 列名

sheetHeadRow.createCell(i - offset).setCellValue(columnName);

}

// 设置数据行

int rowIndex = 1;// 行索引, 第一行为标题行

for (Object tableRow : tableView.getItems()) {

Row sheetDataRow = sheet.createRow(rowIndex++);

for (int i = offset; i < columns.size(); i++) {

String tabelCellData = columns.get(i).getCellData(tableRow).toString();// 数据

sheetDataRow.createCell(i - offset).setCellValue(tabelCellData);// 设置单元格数据

}

try (FileOutputStream fos = new FileOutputStream(file)) {

workbook.write(fos);

}

}

workbook.close();

}

/\*\*

加载excel文件

@param file excel文件, 格式若不正确则抛出异常

\*/

public static List<Student> loadInfo(@NotNull File file) throws IOException {

if (!file.exists() || !file.getName().endsWith("xlsx")) {

throw new IllegalArgumentException("文件格式不正确");

}

List<Student> students = new ArrayList<>();

Workbook workbook = new XSSFWorkbook(new FileInputStream(file));

Sheet sheet = workbook.getSheetAt(0);

Row sheetHead = sheet.getRow(0);

int numCols = sheetHead.getLastCellNum(); // 总行数

int numRows = sheet.getLastRowNum();// 总列数

for (int i = 1; i <= numRows; i++) {

Row row = sheet.getRow(i);

if (row == null) continue;

Student student = createStudent(numCols, row, sheetHead);

students.add(student);

}

workbook.close();

return students;

}

/\*\*

从行中创建一个学生对象

\*/

private static Student createStudent(int numCols, Row row, Row headerRow) {

Student student = new Student();

for (int i = 0; i < numCols; i++) {

Cell cell = row.getCell(i);

if (cell == null) continue;

String value = cell.getStringCellValue();

switch (headerRow.getCell(i).getStringCellValue()) {

// 注意事项: 配置文件中的字段 = excel表的列名 = case项

case "学号" -> student.setId(value);

case "姓名" -> student.setName(value);

case "专业" -> student.setDept(value);

case "班级" -> student.setClassName(value);

case "年龄" -> student.setAge(Integer.parseInt(value));

case "性别" -> student.setSex(value);

case "籍贯" -> student.setNativePlace(value);

case "邮箱" -> student.setEmail(value);

case "出生日期" -> student.setBirthday(cell.getDateCellValue().toInstant()

.atZone(ZoneId.systemDefault()).toLocalDate());

case "政治面貌" -> student.setOutlook(value);

case "家庭住址" -> student.setAddress(value);

case "电话号码" -> student.setPhone(value);

}

}

return student;

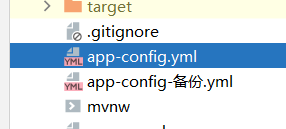
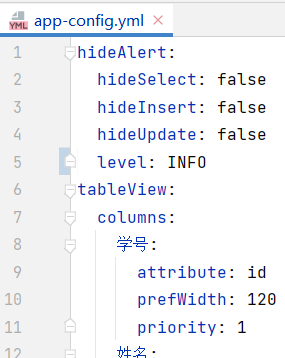
}

private ExcelUtil() {// 无实例

}

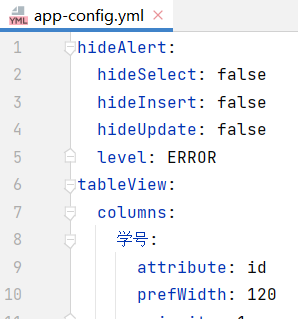
}

### ② 配置文件功能

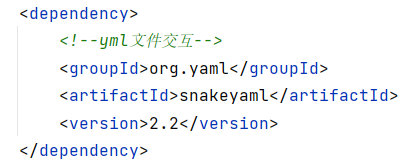
 

设置 – 将弹窗级别从INFO改为ERROR – 再次查看配置文件,成功同步更改





导入依赖项org.yaml.snakeyaml



代码部分:

package com.jjdx.studentmanage.Util;

import org.yaml.snakeyaml.DumperOptions;

import org.yaml.snakeyaml.Yaml;

import java.io.FileInputStream;

import java.io.FileWriter;

import java.util.HashMap;

/\*\*

Yaml配置文件的读取器, (需要在程序启动前加载信息)

@ Author: 绝迹的星 <br>

@ Time: 2024/5/7 <br> \*/

public class YamlUtil {

private static HashMap<String, Object> yMap = new HashMap<>();//从配置文件读取的信息

private static String configFile = "app-config.yml";// 配置文件的文件名

private static final Yaml yaml;

static {

DumperOptions options = new DumperOptions();

options.setDefaultFlowStyle(DumperOptions.FlowStyle.BLOCK);// 块式写入

options.setPrettyFlow(true);

yaml = new Yaml(options);

}

private YamlUtil() {// 无实例

}

/\*\*

项目启动, 读取配置文件, 若无法读取,则强制停止程序运行

\*/

public static void start() {

yMap = reLoad();

if (yMap != null) return;

System.out.println("配置文件加载失败, 尝试读取备份");

configFile = "app-config-备份.yml";

yMap = reLoad();

if (yMap == null) {

throw new NullPointerException("备份文件加载失败, 请检查文件");

}

try {

FileWriter output = new FileWriter("app-config.yml");

yaml.dump(yMap, output);

configFile = "app-config.yml";

} catch (Exception e) {

System.out.println("备份文件读回失败 -- " + e.getMessage());

}

}

/\*\*

重新加载配置文件, 由于实现实时更新的效果, 所以每次都需要重新读入配置文件

@return 配置文件中的信息

\*/

private static HashMap<String, Object> reLoad() {

try {

return yaml.load(new FileInputStream(configFile));

} catch (Exception e) {

e.printStackTrace();

return null;

}

}

/\*\*

分隔获取最后一个HashMap

- 返回的HashMap是可修改的, 不允许外界访问该方法

@param split 键名

@return 最后一个HashMap, 若不存在则返回null

\*/

private static HashMap<String, Object> split(String[] split) {

HashMap<String, Object> m = new HashMap<>(yMap);

int n = split.length;

for (int i = 0; i < n - 1; i++) {

if (!m.containsKey(split[i])) return null;

Object o = m.get(split[i]);

if (!(o instanceof HashMap)) return null;

m = (HashMap<String, Object>) o;

}

return m;

}

/\*\*

获取配置文件的值

@param name 以"."分隔的键名

@return 不存在键时返回默认值

\*/

public static <T> T get(String name, T defaultValue) {

reLoad();

String[] split = name.split("\\.");

HashMap<String, Object> lastMap = split(split);

if (lastMap == null) return defaultValue;

Object value = lastMap.get(split[split.length - 1]);

if (!ClassUtil.isInstanceOf(value, defaultValue.getClass())) {

return defaultValue;

}

return (T) value;

}

/\*\*

获取配置文件的int值

@param name 以"."分隔的键名

@return 值, 若不存在则返回默认值0

\*/

public static int getInt(String name) {

reLoad();

return getInt(name, 0);

}

/\*\*

获取配置文件的int值

@param name 以"."分隔的键名

@param defaultVal 不存在则返回默认值

\*/

public static int getInt(String name, int defaultVal) {

reLoad();

String[] split = name.split("\\.");

HashMap<String, Object> lastMap = split(split);

if (lastMap == null) return defaultVal;

Object value = lastMap.get(split[split.length - 1]);

if (!ClassUtil.isInstanceOf(value, Integer.class)) {

return defaultVal;

}

return (int) value;

}

/\*\*

获取配置文件的String值

@param name 以"."分隔的键名

@return 不存在则返回默认值""

\*/

public static String getString(String name) {

return getString(name, "");

}

/\*\*

获取配置文件的String值

@param name 以"."分隔的键名

@param defaultVal 不存在则返回默认值

\*/

public static String getString(String name, String defaultVal) {

reLoad();

String[] split = name.split("\\.");

HashMap<String, Object> lastMap = split(split);

if (lastMap == null) return defaultVal;

Object value = lastMap.get(split[split.length - 1]);

if (!ClassUtil.isInstanceOf(value, String.class)) {

return defaultVal;

}

return (String) value;

}

/\*\*

获取配置文件的boolean值

@param name 以"."分隔的键名

@return 不存在则返回false

\*/

public static boolean getBoolean(String name) {

reLoad();

String[] split = name.split("\\.");

HashMap<String, Object> lastMap = split(split);

if (lastMap == null) return false;

Object value = lastMap.get(split[split.length - 1]);

if (!ClassUtil.isInstanceOf(value, Boolean.class)) {

return false;

}

return (boolean) value;

}

/\*\*

更改配置文件

@param name 以"."分隔的键名, 必须存在才能更改

@param value 新值

\*/

public static void change(String name, Object value) {

reLoad();

//获取最后一个map

String[] split = name.split("\\.");

HashMap<String, Object> lastMap = split(split);

if (lastMap == null) return;

//设置值

lastMap.put(split[split.length - 1], value);

try {

FileWriter output = new FileWriter(configFile);

yaml.dump(yMap, output);

} catch (Exception e) {

System.out.println("配置文件更改失败 -- " + e.getMessage());

}

}

/\*\*

获取最后一个层级对应的HashMap

@param name 以"."分隔的键名, 应当以"."或".null"结尾,表示层级终止

@return 只读数据

\*/

public static HashMap<String, Object> getLastMap(String name) {

reLoad();

if (!(name.endsWith(".") || name.endsWith(".null"))) name += ".null";

String[] split = name.split("\\.");

HashMap<String, Object> lastMap = split(split);

if (lastMap == null) return null;

return new HashMap<>(lastMap);

}

}

# 六、问题和讨论

## (1) 重复代码

在进行错误信息校验时, 校验项非常多, 并且同一属性校验的方式可能还不一样

比如学号, 在select可以不填, 而在insert中是必填且数据库中不存在的, 在update中又需要是必填且数据库中存在的.

实际上, 其本身的校验逻辑并不复杂, 但是多个数据的多种校验组合使得代码重复冗长, 编写和阅读都比较费力

这种情况不知道有什么好的解决办法 (表驱动? 多态?)

## (2) 路径问题

在代码中需要编写文件相对路径, 路径很容易写错

尤其是使用如JavaFX的框架项目, 文件存放在resources目录下, 并且文件查找规则与类所在的包相关

用绝对路径虽然有效, 但不规范, 因为无法进行代码移植

## (3) 异常处理

在代码中会有许多运行时的异常, 应该选择任其抛出, 终止程序运行, 还是try catch 掉每一个异常, 自行处理?

在更麻烦的一点是, 有时候发生异常的是程序的关键类, 如果在这个类内自行处理异常, 但关键类的数据丢失会导致许多类无法正常运行, 例如我的配置类, 配置类中的信息会给许多类使用, 其中主界面使用了它的大量数据, 如果配置文件无法加载, 程序是无法运行的, 所以如果配置类出错, 我选择了直接中断程序运行

# 七、总结体会

在完成《学生信息管理系统》的程序设计课程报告过程中，我深刻体会到了程序设计的魅力与挑战。本次实验不仅锻炼了我的编程技能，更让我对信息系统的设计与实现有了更加深入的理解。

通过这次实验，我获得了宝贵的学习和成长, 我不仅提升了自己的编程技能，还学会了如何将理论知识应用到实际项目中, 实验的每一个环节都极大地拓宽了我的视野。我相信这些经验将对我未来的学习和职业生涯产生积极的影响, 我期待将这些经验应用到未来的学习和工作中，继续在程序设计的道路上探索和前进。