****



代码清单说明

--爱刷题



小组编号：g003

组长：张浩

组员：金方永，陈紫慧

**版本控制页**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 修订日期 | 版本/状态 | 修订人 | 备注 |
| 1 | 2021/12/5 | 1.0 | 金方永 | 修订代码展示和打包代码作为附件 |
| 2 | 2021/12/5 | 1.1 | 陈紫慧 | 修订 |

目录

[一、项目描述 4](#_Toc89803571)

[1.1标识 4](#_Toc89803572)

[1.2系统概述 4](#_Toc89803573)

[1.4基线 6](#_Toc89803574)

[1.3引用文件 6](#_Toc89803575)

[二、目录结构 7](#_Toc89803576)

## 一、项目描述

## 1.1标识

本条应包含本文档适用的系统和软件的完整标识，(若适用)包括标识号、标题、缩略词语、版本号和发布号。

|  |  |  |
| --- | --- | --- |
| 文件状态 | 文件标识 | SE2021-G003-代码清单 |
| [] 草稿 | 当前版本 | 1.2 |
| [√ ] 正式发布 | 作者 | 张浩，金方永，陈紫慧 |
| [ ] 正在修改 | 编写日期 | 2021/12/25 |

## 1.2系统概述

项目作为软件工程课程项目，旨在用实例驱动，将软件工程总流程在实例中的各阶段得到体现，综合提升软件工程素养。本项目的主题是通过设计一个刷题平台，供城院学生及老师使用，要求通过本平台实现无纸化刷题测试，目标是突破传统纸质考试的弊端，成为现代学校教育重要的辅助工具，通过合理运用在线考试系统能减轻教师工作负担、提升工作效率；学生能实现多次模拟，及时反馈，练习考试，强化学习成果。实现环境是以课程为载体，通过学习相关技术实现课程要求，体现出软件工程的控制流程，完成各项要求。

一般特性：

基于vue、springboot开发的刷题网站

开发、运行和维护的历史：

开发历史

|  |  |
| --- | --- |
| 时间 | 内容 |
| 2021/9/13-2021/9/21 | 项目开发方向及形式确定 |
| 2021/9/23-2021/10/11 | 项目计划 |
| 2021/10/14-2021/10/21 | 可行性分析报告 |
| 2021/10/21-2021/11/2 | 软件需求说明 |
| 2021/11/2-2021/11/12 | 项目设计 |
| 2021/11/12-2021/11/14 | 用户手册 |
| 2021/11/14-2021/11/21 | 详细设计文档 |
| 2021/11/22-2021-11-29 | 测试说明文档 |
| 2021/11/30-2021-12-5 | 代码清单 |

暂无运行和维护的历史

投资方：G003组全体成员

用户：浙大城市学院全体师生

开发方：G003组全体成员

## 1.4基线

1. 项目计划书-SE2021-g003-项目计划.doc
2. 可行性报告-SE2021-G003-可行性报告.doc
3. 软件需求说明-SE2021-G003-需求说明报告.doc
4. 项目设计-数据库（顶层）设计说明.doc
5. 数据字典-爱刷题数据字典.doc
6. 总体设计报告-SE2021-G003-总体设计报告.doc
7. 数据库（顶层）设计说明(DBDD).doc
8. 详细设计文档（SDD）.doc
9. 测试说明文档.doc
10. 代码清单.doc

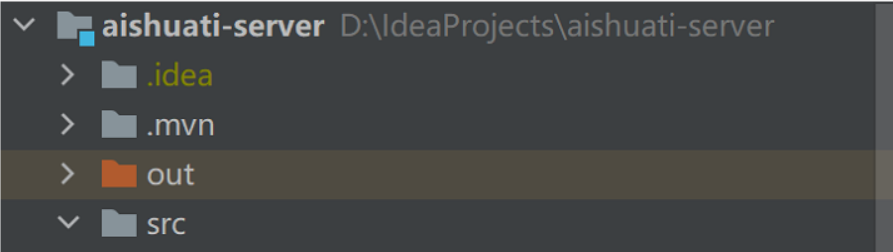
# 1.3引用文件

本章应列出本文档引用的所有文档的编号、标题、修订版本和日期，也应标识不能通过正常的供货渠道获得的所有文档的来源。

1. 《GB T-8567-2006计算机软件文档编制规范》15-软件测试说明（STD）
2. 《软件工程导论（第六版）》张海藩编着清华大学出版社
3. 项目计划书-SE2021-g003-项目计划doc
4. 可行性报告-SE2021-G003-可行性报告doc
5. 软件需求说明-SE2021-G003-需求说明报告doc
6. 项目设计-数据库（顶层）设计说明doc
7. 数据字典-爱刷题数据字典doc
8. 数据库（顶层）设计说明(DBDD).doc
9. 详细设计文档（SDD）.doc
10. 测试说明文档.doc
11. 代码清单.doc

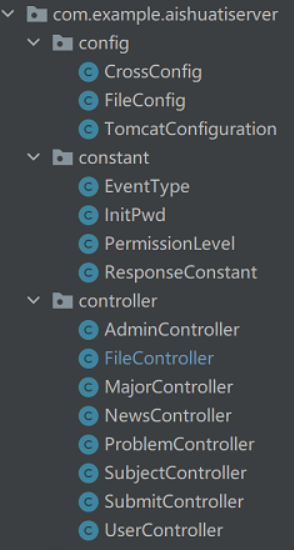
# 二、目录结构

3.1后端目录结构说明



说明…

|  |  |
| --- | --- |
| 目录 | 说明 |
| Config | 用于存放配置类 |
| Constant | 不用存放不变的信息 |
| Controller | 控制层，用于与前端交互 |
| JavaBean | 存放实体类 |
| Mapper | 用于映射，有关数据库 |
| Service | 服务层 |
| Util | 存放公共类，降低代码耦合度 |



说明：

config：

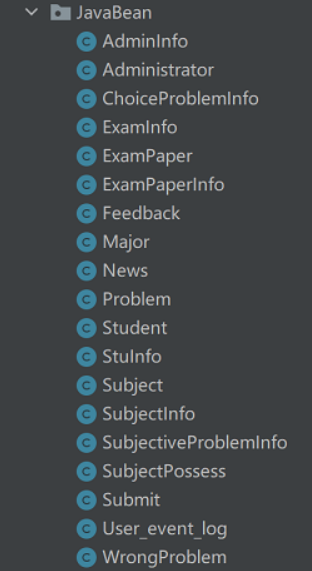
1.CrossConfig类用于配置服务器跨域的问题

2.FileConfig类用于文件上传和下载服务器地址的配置，便于Service层自动装配

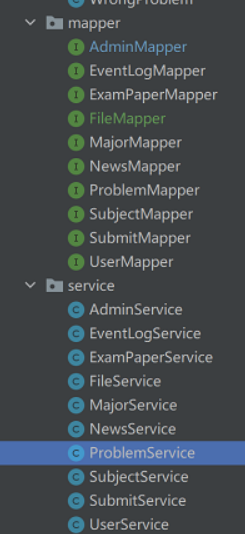
3.TomcatConfiguration类用于配置服务器session存储的问题，不配置会导致登录之后会话不一致。

constant：

1. EventType：事件种类，用于存储日志，登录还是登出
2. InitPwd：管理员添加教师用户时的默认密码
3. PermissionLevel：用于区分管理员和教师的权限，管理员权限999，教师权限为0
4. ResponseConstant：返回简单Map<String,Object> 给前端，包括一些简单的消息



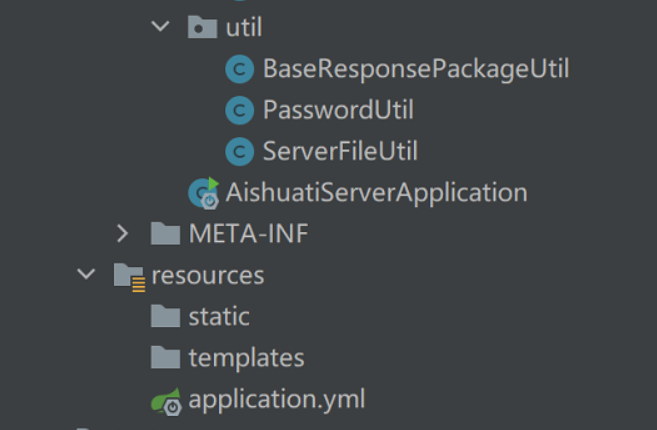
说明 项目中的实体类



说明

mapper：

1. AdminMapper：用于编写有关管理员的sql语句
2. EventLogMapper：用于编写有关日志信息的sql语句
3. ExamPaperMapper：用于编写有关试卷信息的sql语句
4. FileMapper：用于编写文件信息的sql语句
5. MajorMapper：用于编写有关专业信息的sql语句
6. NewsMapper：用于编写有关公告信息的sql语句
7. problemMapper：用于编写有关题目信息的sql语句
8. SubjectMapper：用于编写学科有关信息的sql语句
9. SubmitMapper：用于学生用户提交答案生成记录
10. UserMaper：用于有关学生用户信息的sql语句



Util：

1. BaseResponsePackage：包封装了ImmutableMap类，用于返回前端信息
2. PasswordUtil：用于密码的生成和校验
3. ServerFileUtil：用于服务器文件地址的生成和的得到完整路径

代码清单展示：

AdminController：

package com.example.aishuatiserver.controller;  
  
import com.alibaba.fastjson.JSON;  
import com.alibaba.fastjson.JSONObject;  
import com.example.aishuatiserver.JavaBean.Administrator;  
import com.example.aishuatiserver.constant.InitPwd;  
import com.example.aishuatiserver.constant.PermissionLevel;  
import com.example.aishuatiserver.constant.ResponseConstant;  
import com.example.aishuatiserver.service.AdminService;  
import com.example.aishuatiserver.service.MajorService;  
import com.example.aishuatiserver.service.UserService;  
import com.example.aishuatiserver.util.BaseResponsePackageUtil;  
import com.google.common.collect.ImmutableMap;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
  
import javax.servlet.http.HttpServletRequest;  
import java.util.Map;  
  
@RestController  
@RequestMapping(value = "admin")  
public class AdminController {  
  
 @Autowired  
 private AdminService adminService;  
  
 @Autowired  
 private MajorService majorService;  
  
 @Autowired  
 private UserService userService;  
  
 @RequestMapping(value = "/reg", method = RequestMethod.POST)  
 public Map<String, Object> reg(  
 @RequestBody JSONObject p,  
 HttpServletRequest request  
 ) {  
 String adminAccount = p.getString("adminAccount");  
 String adminName = p.getString("adminName");  
 String adminEmail = p.getString("adminEmail");  
 String adminTelephoto = p.getString("adminTelephoto");  
 if (!adminService.checkPermission(request.getSession(), PermissionLevel.SUPER\_ADMIN)){  
 return ResponseConstant.X\_ACCESS\_DENIED;  
 }  
 String pwd1 = InitPwd.INITIAL\_PWD;  
 adminService.reg(adminAccount, adminName, pwd1, adminEmail, adminTelephoto);  
 return ResponseConstant.V\_ADD\_SUCCESS;  
 }  
  
 @RequestMapping(value = "/login", method = RequestMethod.POST)  
 public Map<String, Object> login(  
 @RequestBody JSONObject p,  
 HttpServletRequest request  
 ) {  
 String adminAccount = p.getString("adminAccount");  
 String adminPwd = p.getString("adminPwd");  
 Administrator admin = adminService.getAdminByAdminAccount(adminAccount);  
 if (admin == null) {  
 return ResponseConstant.X\_USER\_NOT\_FOUND;  
 }  
 if (!adminService.checkPwd(admin, adminPwd)) {  
 return ResponseConstant.X\_USER\_WRONG\_PASSWORD;  
 }  
 adminService.savaAdminToSession(request.getSession(), admin);  
 return ResponseConstant.V\_USER\_LOGIN\_SUCCESS;  
 }  
  
 @RequestMapping(value = "/logout", method = RequestMethod.POST)  
 public Map<String, Object> logout(  
 HttpServletRequest request  
 ) {  
 adminService.removeAdminFromSession(request.getSession());  
 return ResponseConstant.V\_USER\_LOGOUT\_SUCCESS;  
 }  
  
 @RequestMapping(value = "/updateState/{adminId}", method = RequestMethod.POST)  
 public Map<String, Object> updateState(  
 @PathVariable(value = "adminId") int adminId,  
 HttpServletRequest request  
 ) {  
 if (!adminService.checkPermission(request.getSession(), PermissionLevel.SUPER\_ADMIN)) {  
 return ResponseConstant.X\_ACCESS\_DENIED;  
 }  
 adminService.updateState(adminId);  
 return ResponseConstant.V\_UPDATE\_SUCCESS;  
 }  
  
 @RequestMapping(value = "/getAdminByAdminId/{adminId}",method = RequestMethod.POST)  
 public Map<String,Object> getAdminByAdminId(  
 @PathVariable(value = "adminId") int adminId  
 ){  
 return BaseResponsePackageUtil.baseData(adminService.getAdminByAdminId(adminId));  
 }  
  
 @RequestMapping(value = "/updateMyPwd", method = RequestMethod.POST)  
 public Map<String, Object> updateMyPwd(  
 @RequestBody JSONObject p,  
 HttpServletRequest request  
 ) {  
 String pwd1 = p.getString("pwd1");  
 String pwd2 = p.getString("pwd2");  
 if (!pwd1.equals(pwd2)) {  
 return ResponseConstant.X\_REPLACE\_PWD;  
 }  
 int adminId = adminService.getAdminIdFromSession(request.getSession());  
 String adminAccount = adminService.getAdminAccountFromSession(request.getSession());  
 adminService.updateMyPwd(pwd1, adminAccount, adminId);  
 return ResponseConstant.V\_UPDATE\_SUCCESS;  
 }  
  
 @RequestMapping(value = "/updateMyInfo", method = RequestMethod.POST)  
 public Map<String, Object> updateMyInfo(  
 @RequestBody JSONObject p,  
 HttpServletRequest request  
 ) {  
 String email = p.getString("email");  
 String telephoto = p.getString("adminTelephoto");  
 String introduce = p.getString("introduce");  
 int adminId = adminService.getAdminIdFromSession(request.getSession());  
 adminService.updateMyInfo(email, telephoto, introduce, adminId);  
 return ResponseConstant.V\_UPDATE\_SUCCESS;  
 }  
  
 @RequestMapping(value = "/getAllAdminInfo/{page}/{pageSize}",method = RequestMethod.POST)  
 public Map<String,Object> getAllAdminInfo(  
 @PathVariable(value = "page") int page,  
 @PathVariable(value = "pageSize") int pageSize,  
 HttpServletRequest request  
 ){  
 return BaseResponsePackageUtil.baseData(  
 ImmutableMap.of(  
 "total",adminService.getAdminCount(),  
 "rows",adminService.getAllAdminInfo(page,pageSize)  
 ));  
 }  
  
 @RequestMapping(value = "/changeTeacherInfo",method = RequestMethod.POST)  
 public Map<String,Object> changeAdminInfo(  
 @RequestBody JSONObject p,  
 HttpServletRequest request  
 ){  
 int adminId = p.getInteger("adminId");  
 String adminName = p.getString("adminName");  
 String adminEmail = p.getString("adminEmail");  
 String adminTelephone = p.getString("adminTelephone");  
 String adminAccount = p.getString("adminAccount");  
 if (!adminService.checkPermission(request.getSession(), PermissionLevel.SUPER\_ADMIN)) {  
 return ResponseConstant.X\_ACCESS\_DENIED;  
 }  
 adminService.changeTeacherInfo(adminId,adminName,adminEmail,adminTelephone,adminAccount);  
 return ResponseConstant.V\_UPDATE\_SUCCESS;  
 }  
  
 @RequestMapping(value = "/changeStuInfoById",method = RequestMethod.POST)  
 public Map<String,Object> changeStuInfo(  
 @RequestBody JSONObject p,  
 HttpServletRequest request  
 ){  
 if (!adminService.checkPermission(request.getSession(), PermissionLevel.SUPER\_ADMIN)) {  
 return ResponseConstant.X\_ACCESS\_DENIED;  
 }  
 int stuId = p.getInteger("stuId");  
 String stuName = p.getString("stuName");  
 String stuNickName = p.getString("stuNickName");  
 String stuEmail = p.getString("stuEmail");  
 String stuTelephoto = p.getString("stuTelephoto");  
 String majorName = p.getString("majorName");  
 String stu\_level = p.getString("stu\_level");  
 int majorId = majorService.getMajorIdByMajorName(majorName);  
 if(majorId == 0){  
 return ResponseConstant.X\_MAJOR\_NOT\_FIND;  
 }  
 userService.adminChangeStuInfo(stuId,stuName,stuNickName,stuEmail,stuTelephoto,majorId,stu\_level);  
 return ResponseConstant.V\_UPDATE\_SUCCESS;  
 }  
}

package com.example.aishuatiserver.controller;  
  
import com.alibaba.fastjson.JSONObject;  
import com.example.aishuatiserver.config.FileConfig;  
import com.example.aishuatiserver.constant.ResponseConstant;  
import com.example.aishuatiserver.service.AdminService;  
import com.example.aishuatiserver.service.ExamPaperService;  
import com.example.aishuatiserver.service.FileService;  
import com.example.aishuatiserver.service.UserService;  
import com.example.aishuatiserver.util.BaseResponsePackageUtil;  
import com.example.aishuatiserver.util.HttpUtil;  
import com.google.common.collect.ImmutableMap;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.core.io.FileSystemResource;  
import org.springframework.http.MediaType;  
import org.springframework.http.MediaTypeFactory;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
import org.springframework.web.multipart.MultipartFile;  
  
import javax.servlet.http.HttpServletRequest;  
import java.io.File;  
import java.util.Map;  
  
@RestController  
@RequestMapping(value = "/examPaper")  
public class ExamPaperController {  
  
 @Autowired  
private FileService fileService;  
  
 @Autowired  
private AdminService adminService;  
  
 @Autowired  
private ExamPaperService examPaperService;  
  
 @Autowired  
private UserService userService;  
  
 @RequestMapping(value = "/download", method = RequestMethod.POST)  
public ResponseEntity<FileSystemResource> download(  
 @RequestBody JSONObject p  
 ) {  
 String filename = p.getString("filename");  
 if (filename == null) {  
 return ResponseEntity.badRequest().build();  
 }  
 File file = new File(fileService.getFilePath(filename));  
 if (!file.exists()) {  
 return ResponseEntity.notFound().build();  
 }  
 MediaType mediaType = MediaType.parseMediaType("application/octet-stream");  
 if (MediaTypeFactory.getMediaType(filename).isPresent())  
 mediaType = MediaTypeFactory.getMediaType(filename).get();  
 return ResponseEntity  
 .ok()  
 .headers(HttpUtil.fileHeadersUUID(filename))  
 .contentLength(file.length())  
 .contentType(mediaType)  
 .body(new FileSystemResource(file));  
 }  
  
 @RequestMapping(value = "upload", method = RequestMethod.POST)  
public Map<String,Object> upload(  
 @RequestParam(value = "subjectName") String subjectName,  
 @RequestParam(value = "examPaperName") String examPaperName,  
 @RequestParam(value = "level") String level,  
 @RequestParam(value = "examPaperFrom") String examPaperFrom,  
 @RequestParam(value = "file") MultipartFile file,  
 HttpServletRequest request  
 ){  
 try {  
 int adminId = adminService.getAdminIdFromSession(request.getSession());  
 String examPaperAddress = fileService.newFile(file);  
 examPaperService.addExamPaper(subjectName,examPaperName,level,examPaperFrom,examPaperAddress,adminId);  
 return ResponseConstant.V\_ADD\_SUCCESS;  
 }catch (Exception e){  
 e.printStackTrace();  
 return ResponseConstant.X\_SYSTEM\_ERROR;  
 }  
 }  
  
 @RequestMapping(value = "/showAllExamPaper/{page}/{size}")  
public Map<String,Object> getAllExamPaper(  
 @PathVariable(value = "page") int page,  
 @PathVariable(value = "size") int size,  
 HttpServletRequest request  
 ){  
 Integer adminId = adminService.getAdminIdFromSession(request.getSession());  
 if(adminId == null) return ResponseConstant.X\_USER\_LOGIN\_FIRST;  
 return BaseResponsePackageUtil.baseData(  
 ImmutableMap.of(  
 "total",examPaperService.getAllExamPaperCount(),  
 "rows",examPaperService.getAllExamPaper(page,size)  
 ));  
 }  
  
 @RequestMapping(value = "/getMyExamPaper",method = RequestMethod.POST)  
public Map<String,Object> stuGetMyExamPaper(  
 @RequestBody JSONObject p,  
 HttpServletRequest request  
 ){  
 int page = p.getInteger("page");  
 int pageSize = p.getInteger("pageSize");  
 int stuId = userService.getStuIdBySession(request.getSession());  
 return BaseResponsePackageUtil.baseData(  
 ImmutableMap.of(  
 "total",examPaperService.stuGetMyExamPaperCount(stuId),  
 "data",examPaperService.stuGetMyExamPaper(stuId,page,pageSize)  
 ));  
 }  
  
 @RequestMapping(value = "/search/stu",method = RequestMethod.POST)  
public Map<String,Object> stuSearchExamPaper(  
 @RequestBody JSONObject p,  
 HttpServletRequest request  
 ){  
 int page = p.getInteger("page");  
 int pageSize = p.getInteger("pageSize");  
 String subjectName = p.getString("subjectName");  
 Integer examPaperId = p.getInteger("examPaperId");  
 int stuId = userService.getStuIdBySession(request.getSession());  
 if(examPaperId == 0) examPaperId = null;  
 return BaseResponsePackageUtil.baseData(  
 ImmutableMap.of(  
 "total",examPaperService.stuSearchMyExamPaperCount(stuId,subjectName,examPaperId),  
 "data",examPaperService.stuSearchMyExamPaper(stuId,subjectName,examPaperId,page,pageSize)  
 ));  
 }  
  
}

package com.example.aishuatiserver.controller;  
  
  
import com.alibaba.fastjson.JSONObject;  
import com.example.aishuatiserver.constant.PermissionLevel;  
import com.example.aishuatiserver.constant.ResponseConstant;  
import com.example.aishuatiserver.service.AdminService;  
import com.example.aishuatiserver.service.EventLogService;  
import com.example.aishuatiserver.util.BaseResponsePackageUtil;  
import com.google.common.collect.ImmutableMap;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.RequestBody;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RequestMethod;  
import org.springframework.web.bind.annotation.RestController;  
  
import javax.servlet.http.HttpServletRequest;  
import java.util.Map;  
  
@RestController  
@RequestMapping(value = "logs")  
public class LogController {  
  
 @Autowired  
private EventLogService eventLogService;  
  
 @Autowired  
private AdminService adminService;  
  
 @RequestMapping(value = "/get",method = RequestMethod.POST)  
public Map<String,Object> getAllLogs(  
 @RequestBody JSONObject p,  
 HttpServletRequest request  
 ){  
 int page = p.getInteger("page");  
 int pageSize = p.getInteger("pageSize");  
 if(adminService.checkPermission(request.getSession(), PermissionLevel.SUPER\_ADMIN)){  
 return ResponseConstant.X\_ACCESS\_DENIED;  
 }  
 return BaseResponsePackageUtil.baseData(  
 ImmutableMap.of(  
 "total",eventLogService.getAllLogCount(),  
 "data",eventLogService.getUserLogs(page,pageSize)  
 ));  
 }  
  
  
  
}

package com.example.aishuatiserver.mapper;  
  
  
import com.example.aishuatiserver.JavaBean.ChoiceProblemInfo;  
import com.example.aishuatiserver.JavaBean.SubjectiveProblemInfo;  
import com.example.aishuatiserver.JavaBean.WrongProblem;  
import org.apache.ibatis.annotations.Insert;  
import org.apache.ibatis.annotations.Mapper;  
import org.apache.ibatis.annotations.Select;  
  
import java.util.List;  
  
@Mapper  
public interface ProblemMapper {  
  
 @Select("select count(\*) from problem where problem\_type = 1 and subject\_Id in (select subject\_Id from possesse where stu\_Id = #{stuId})")  
 int getMyAllChoiceProblemCount(int stuId);  
  
 @Select("select problem\_Id as problemId,subject\_Name as subjectName,Administrator\_id as administratorId,difficult,info\_text\_content,correct as ans,choice\_A,choice\_B,choice\_C,choice\_D,reference " +  
 "from problem left join (select stu\_Id,possesse.subject\_Id,subject\_name from possesse left join subject on possesse.subject\_id = `subject`.subject\_Id ) as a on problem.subject\_id = a.subject\_Id " +  
 "where problem\_type = 1 and stu\_id = #{stuId} limit #{offSize},#{size}")  
 List<ChoiceProblemInfo> getAllChoiceProblemInfo(int stuId,int offSize,int size);  
  
 @Select("select problem\_Id as problemId,subject\_Name as subjectName,Administrator\_id as administratorId,difficult,info\_text\_content,correct as ans,choice\_A,choice\_B,choice\_C,choice\_D,reference " +  
 "from problem left join subject on problem.subject\_Id = subject.subject\_Id where problem\_type = 1 order by problem.subject\_Id limit #{offset},#{size}")  
 List<ChoiceProblemInfo> adminGetAllChoiceProblem(int offset,int size);  
  
 @Select("select problem\_Id as problemId,subject\_Name as subjectName,Administrator\_id as administratorId,difficult,info\_text\_content,reference " +  
 "from problem left join subject on problem.subject\_Id = subject.subject\_Id where problem\_type = 2 order by problem.subject\_Id limit #{offset},#{size}")  
 List<SubjectiveProblemInfo> adminGetAllSubjectiveProblem(int offset,int size);  
  
 @Select("select count(\*) from problem where problem\_type = 2")  
 int adminGetAllSubjectiveProblemCount();  
  
 @Select("select count(\*) from problem where problem\_type = 1")  
 int adminGetAllChoiceProblemCount();  
  
 @Select("select count(\*) from problem where problem\_type = 2 and subject\_Id in (select subject\_Id from possesse where stu\_Id = #{stuId})")  
 int getMyAllSubjectiveProblemCount(int stuId);  
  
 @Select("select problem\_Id as problemId,subject\_Name as subjectName,Administrator\_id as administratorId,difficult,info\_text\_content,reference " +  
 "from problem right join (select stu\_Id,possesse.subject\_Id,subject\_name from possesse left join subject on possesse.subject\_id = subject.subject\_Id where stu\_id = #{stuId}) as a on problem.subject\_id = a.subject\_Id " +  
 "where problem\_type = 2 and stu\_id = #{stuId} limit #{offSize},#{size}")  
 List<SubjectiveProblemInfo> getAllSubjectiveProblemInfo(int stuId,int offSize,int size);  
  
 @Select("<script> " +  
 "select count(\*) from problem where problem\_type = 1 " +  
 "<if test = \"subjectId != null\">and subject\_Id = #{subjectId} </if>" +  
 "<if test = \"problemId != null\">and problem\_Id = #{problemId}</if>" +  
 "</script>")  
 int getSearchChoiceCount(int stuId,Integer subjectId,Integer problemId);  
  
 @Select("<script> " +  
 "select count(\*) from problem where problem\_type = 2 " +  
 "<if test = \"subjectId != null\">and subject\_Id = #{subjectId} </if>" +  
 "<if test = \"problemId != null\">and problem\_Id = #{problemId}</if>" +  
 "</script>")  
 int getSearchSubjectiveCount(int stuId,Integer subjectId,Integer problemId);  
  
 @Select("<script>" +  
 "select problem\_Id as problemId,subject\_Name as subjectName,Administrator\_id as administratorId,difficult,info\_text\_content,correct as ans,choice\_A,choice\_B,choice\_C,choice\_D,reference "+  
 "from problem inner join (select stu\_Id,possesse.subject\_Id,subject\_name from possesse left join subject on possesse.subject\_id = subject.subject\_Id where stu\_id = #{stuId}) as a on problem.subject\_id = a.subject\_Id "+  
 "where problem\_type = 1 "+  
 "<if test = \"subjectName != null \" >"+  
 " and subject\_Name = #{subjectName} "+  
 "</if>"+  
 "<if test = \"problemId != null\">"+  
 "and problem\_Id = #{problemId} "+  
 "</if>"+  
 "limit #{offset},#{size}"+  
 "</script>")  
 List<ChoiceProblemInfo> searchChoiceProblemInfo(int stuId,String subjectName,Integer problemId,int offset,int size);  
  
 @Select("select IFNULL(max(problem\_Id),0) from problem")  
 int getMaxProblemId();  
  
 @Insert("insert into problem (problem\_Id,subject\_Id,Administrator\_id,problem\_type,difficult,info\_text\_content,correct,choice\_A,choice\_B,choice\_C,choice\_D,reference) values(#{problemId},#{subjectId},#{adminId},#{type},#{difficult},#{info\_text\_content},#{correct},#{choice\_A},#{choice\_B},#{choice\_C},#{choice\_D},#{reference})")  
 void addChoiceProblem(int problemId, int subjectId, int adminId, int type, int difficult, String info\_text\_content, int correct, String choice\_A, String choice\_B, String choice\_C, String choice\_D, String reference);  
  
 @Insert("insert into problem (problem\_Id,subject\_Id,Administrator\_id,problem\_type,difficult,info\_text\_content,reference) values(#{problemId},#{subjectId},#{adminId},#{type},#{difficult},#{info\_text\_content},#{reference})")  
 void addSubjectiveProblem(int problemId, int subjectId, int adminId, int type, int difficult, String info\_text\_content, String reference);  
  
 @Select("<script>" +  
 "select problem\_Id as problemId,subject\_Name as subjectName,Administrator\_id as administratorId,difficult,info\_text\_content,reference "+  
 "from problem inner join (select stu\_Id,possesse.subject\_Id,subject\_Name from possesse left join subject on possesse.subject\_id = subject.subject\_Id where stu\_id = #{stuId} ) as a on problem.subject\_id = a.subject\_Id "+  
 "where problem\_type = 2 "+  
 "<if test = \"subjectName != null \" >"+  
 " and subject\_Name = #{subjectName} "+  
 "</if>"+  
 "<if test = \"problemId != null\">"+  
 "and problem\_Id = #{problemId} "+  
 "</if>"+  
 "limit #{offset},#{size}"+  
 "</script>")  
 List<SubjectiveProblemInfo> searchSubjectiveProblemInfo(int stuId,String subjectName,Integer problemId,int offset,int size);  
  
 @Select("select correct from problem where problem\_Id = #{problemId}")  
 int showThisProblemAns(int problemId);  
  
 @Select("select wrongproblem\_Id,a.problem\_Id as problemId,subject\_Name as subjectName,Administrator\_id as administratorId,difficult,\n" +  
 "info\_text\_content,correct as ans,lastSubmit,choice\_A,choice\_B,choice\_C,choice\_D,reference from\n" +  
 "((select wrongProblem\_Id,problem\_Id,stu\_Id,mySubmit as lastSubmit from wrongProblem where stu\_Id = #{stuId} and wrongProblem\_Id in \n" +  
 "(select max(wrongProblem\_Id) from wrongProblem where stu\_Id = #{stuId} group by problem\_Id)) as a left join \n" +  
 "(select problem\_Id,subject\_Name,Administrator\_id,difficult,info\_text\_content,correct,choice\_A,choice\_B,choice\_C,choice\_D,reference " +  
 "from problem left join `subject` on `subject`.subject\_Id = problem.subject\_Id) as b on a.problem\_Id = b.problem\_Id) limit #{offset},#{size}")  
 List<WrongProblem> getWrongProblem(int stuId,int offset,int size);  
  
 @Select("<script>" +  
 "select wrongproblem\_Id,a.problem\_Id as problemId,subject\_Name as subjectName,Administrator\_id as administratorId,difficult,\n" +  
 "info\_text\_content,correct as ans,lastSubmit,choice\_A,choice\_B,choice\_C,choice\_D,reference from\n" +  
 "((select wrongProblem\_Id,problem\_Id,stu\_Id,mySubmit as lastSubmit from wrongProblem where stu\_Id = #{stuId} and wrongProblem\_Id in \n" +  
 "(select max(wrongProblem\_Id) from wrongProblem where stu\_Id = #{stuId} group by problem\_Id)) as a left join \n" +  
 "(select problem\_Id,subject\_Name,Administrator\_id,difficult,info\_text\_content,correct,choice\_A,choice\_B,choice\_C,choice\_D,reference from problem left join `subject` on `subject`.subject\_Id = problem.subject\_Id) as b on a.problem\_Id = b.problem\_Id) "+  
 "<where>" +  
 "<if test = \"subjectName!=null\">subject\_Name = #{subjectName} </if>"+  
 "<if test = \"problemId!=null\"> and a.problem\_Id = #{problemId} </if>"+  
 "</where>"+  
 " limit #{offset},#{size}"+  
 "</script>")  
 List<WrongProblem> searchWrongProblem(int stuId,String subjectName,Integer problemId,int offset,int size);  
  
 @Select("select count(\*) from (select problem\_Id from wrongProblem where stu\_Id = #{stuId} GROUP BY problem\_Id) as a")  
 int getMyWrongProblemCount(int stuId);  
  
 @Select("<script>" +  
 "select count(\*) from (select wrongProblem.problem\_Id,count(\*) from wrongProblem left join problem on wrongProblem.problem\_Id = problem.problem\_Id " +  
 "where stu\_Id = #{stuId} " +  
 "<if test = \"subjectId != null\"> and subject\_Id = #{subjectId} </if>" +  
 "<if test = \"problemId != null\"> and wrongProblem.problem\_Id = #{problemId} </if>"+  
 "</script>")  
 int SearchMyWrongProblemCount(int stuId,Integer subjectId,Integer problemId);  
  
 @Select("<script>"+  
 "select problem\_Id as problemId,subject\_Name as subjectName,Administrator\_id as administratorId,difficult,info\_text\_content,correct as ans,choice\_A,choice\_B,choice\_C,choice\_D,reference " +  
 "from problem left join subject on problem.subject\_Id = subject.subject\_Id where problem\_type = 1 " +  
 "<if test = \"subjectName != null\"> and subject\_Name = #{subjectName} </if> " +  
 "<if test = \"problemId != null\"> and problem.problem\_Id = #{problemId} </if> " +  
 " limit #{offset},#{size}" +  
 "</script>")  
 List<ChoiceProblemInfo> adminSearchChoiceProblem(String subjectName,Integer problemId,int offset,int size);  
  
 @Select("<script> "+  
 "select count(\*) from problem problem left join subject on problem.subject\_Id = subject.subject\_Id where problem\_type = 1 " +  
 "<if test = \"subjectName != null\"> and subject\_Name = #{subjectName} </if> " +  
 "<if test = \"problemId != null\"> and problem\_Id = #{problemId} </if> " +  
 "</script>")  
 int adminSearchChoiceProblemCount(String subjectName,Integer problemId);  
  
 @Select("<script> " +  
 "select problem\_Id as problemId,subject\_Name as subjectName,Administrator\_id as administratorId,difficult,info\_text\_content,reference " +  
 "from problem left join subject on problem.subject\_Id = subject.subject\_Id where problem\_type = 2 " +  
 "<if test = \"subjectName != null\"> and subject\_Name = #{subjectName} </if> " +  
 "<if test = \"problemId != null\"> and problem\_Id = #{problemId} </if> " +  
 " limit #{offset},#{size}" +  
 "</script>")  
 List<SubjectiveProblemInfo> adminSearchSubjectiveProblem(String subjectName,Integer problemId,int offset,int size);  
  
 @Select("<script> "+  
 "select count(\*) from problem problem left join subject on problem.subject\_Id = subject.subject\_Id where problem\_type = 2 " +  
 "<if test = \"subjectName != null\"> and subject\_Name = #{subjectName} </if> " +  
 "<if test = \"problemId != null\"> and problem\_Id = #{problemId} </if> " +  
 "</script>")  
 int adminSearchSubjectiveProblemCount(String subjectName,Integer problemId);  
  
 @Select("select problem\_Id as problemId,subject\_Name as subjectName,Administrator\_id as administratorId,difficult,info\_text\_content,correct as ans,choice\_A,choice\_B,choice\_C,choice\_D,reference " +  
 "from problem left join subject on problem.subject\_Id = subject.subject\_Id where problem\_type = 1 and problem\_Id = #{problemId}")  
 ChoiceProblemInfo getChoiceProblemByProblemId(int problemId);  
  
 @Select("select problem\_Id as problemId,subject\_Name as subjectName,Administrator\_id as administratorId,difficult,info\_text\_content,reference " +  
 "from problem left join subject on problem.subject\_Id = subject.subject\_Id where problem\_type = 2 and problem\_Id = #{problemId}")  
 SubjectiveProblemInfo getSubjectiveProblemByProblem(int problemId);  
}

package com.example.aishuatiserver.mapper;  
  
  
import com.example.aishuatiserver.JavaBean.Subject;  
import com.example.aishuatiserver.JavaBean.SubjectInfo;  
import org.apache.ibatis.annotations.Insert;  
import org.apache.ibatis.annotations.Mapper;  
import org.apache.ibatis.annotations.Select;  
  
import java.util.List;  
  
@Mapper  
public interface SubjectMapper {  
  
 @Select("select IFNULL(max(subject\_Id),0) from subject")  
 int getMaxSubjectId();  
  
 @Select("select count(\*) from subject where subject\_id in (select subject\_id from `subject` where major\_id in (select stu\_majorId from\n" +  
 " student where stu\_Id = #{stu}) and subject\_level in (select stu\_level from student where stu\_Id = #{stu}))")  
 int getAvailableSubjectCount(int stuId);  
  
 @Insert("insert into subject values(#{subjectId},#{majorId},#{subjectName},#{subjectLevel})")  
 void addSubject(Subject subject);  
  
 @Select("select subject\_Id as subjectId,major\_Id as majorId,subject\_Name as subjectName,subject\_level as subjectLevel from subject where subject\_Name=#{subjectName} and major\_Id = #{majorId}")  
 Subject isHad(String subjectName,int majorId);  
  
 @Select("select subjectId,majorName,subjectName,subjectLevel from (select subject\_Id as subjectId,marjor\_Name as majorName,subject\_Name as subjectName,subject\_level as subjectLevel from `subject` left join major on `subject`.major\_Id = major.major\_Id) as b\n" +  
 "left join (select stu\_Id,marjor\_Name,stu\_level FROM student left join major on student.stu\_majorId = major.major\_Id where stu\_Id = #{stuId}) as a on b.majorName = a.marjor\_Name limit #{offset},#{size}")  
 List<SubjectInfo> showSubjectList(int stuId,int offset,int size);  
  
 @Select("select subject\_Id as subjectId,major\_Id as majorId,subject\_Name as subjectName,subject\_level as subjectLevel from subject where \n" +  
 "subject\_Id in (select subject\_Id from possesse where stu\_Id = #{stuId}) limit #{offset},#{size}")  
 List<SubjectInfo> showMySelectSubject(int stuId,int offset,int size);  
  
 @Select("select count(\*) from subject where subject\_Id in (select subject\_id from possesse where stu\_id = #{stuId})")  
 int getMySelectCount(int stu);  
  
 @Insert("insert into possesse values(#{stuId},#{subjectId})")  
 void choiceSubject(int stuId, int subjectId);  
  
 @Select("select subject\_Id from subject where subject\_Name = #{subjectName}")  
 Integer getSubjectIdByName(String subjectName);  
}

3.2 前端学生端目录结构：

|  |  |
| --- | --- |
| node\_modules // 当前项目所有依赖，一般不可以移植给其他电脑环境  public  favicon.ico // 标签图标  index.html // 当前项目唯一的页面  src  assets // 静态资源img、css、js  components // 小组件  views // 页面组件  App.vue // 根组件   main.js // 全局脚本文件（项目的入口）  router 配置路由 |  |
| assets // 静态资源img、css、js |  |
| MainPage:用户进入页面的主界面  PersonalPage：用户个人中心  Login：登陆界面  Regist：注册界面 |  |
| Exampaper：试卷模块  Exerci\_more:多选题  Exerci\_sigle：单选题  Exerci\_subjective：主观题  Home：用户进入的主界面  allNews:：用户所有信息界面  News：信息界面  Welcome：用户进入的欢迎界面  WrongQuestionBook：错题本界面 |  |

3.3 前端管理员端目录结构：

|  |  |
| --- | --- |
| 目录 | 说明 |
| Public：存放图标，公共配置类的文件  Src：业务实现主文件  Env.development：开发环境配置  Env.production：生产环境配置  Eslintrc.js：代码规范格式  Vue.config.js：Vue配置 |  |
| Api：接口配置  Assets：静态文件如图标等  Components：组件  Utils：包装请求类  Views：模块实现  App.vue ：根组件   main.js ：全局脚本文件  router ：配置路由 |  |
| Goods.js：教师接口  Login.js：登陆接口  Member.js：学生接口  Paper.js：试卷接口  Supplier.js：题目接口 |  |
| AppHeader;头部组件  AppMain：主页面  AppNavbar：导航栏  Layout.vue：布局总组件 |  |
| Goods：教师页面  Home：首页  Login：登陆  Member：学生页面  Paper：试卷页面  Supplier：试题页面 |  |