请分析MyInterceptor.java

package org.example.config;

import com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;

import org.example.domain.Sign;

import org.example.service.SignService;

import org.jetbrains.annotations.NotNull;

import org.springframework.stereotype.Component;

import org.springframework.web.servlet.HandlerInterceptor;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.nio.charset.StandardCharsets;

/\*\*

\* @author zjy

\*/

@Component

public class MyInterceptor implements HandlerInterceptor {

private final SignService signService;

public MyInterceptor(SignService signService) {

this.signService = signService;

}

@Override

public boolean preHandle(HttpServletRequest request, @NotNull HttpServletResponse response, @NotNull Object handler)

throws Exception {

String requestRemoteAddr = request.getRemoteAddr();

String url = request.getRequestURI();

if(url.contains(("/sign"))){

Sign one = signService.getOne(new LambdaQueryWrapper<Sign>()

.eq(Sign::getStatus, "已签到")

.eq(Sign::getIp, requestRemoteAddr));

if(one != null) {

response.setContentType("application/json");

response.setCharacterEncoding(StandardCharsets.UTF\_8.toString());

response.getWriter().write("已签到");

response.flushBuffer();

return false;

}

}

return true;

}

}

package org.example.config;

import cn.hutool.extra.qrcode.QrConfig;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import java.awt.\*;

/\*\*

\* @author zjy

\*/

@Configuration

public class QrCode {

@Bean

public QrConfig qrConfig(){

cn.hutool.extra.qrcode.QrConfig qrConfig = new cn.hutool.extra.qrcode.QrConfig();

qrConfig.setBackColor(Color.white.getRGB());//设置背景颜色

qrConfig.setForeColor(Color.black.getRGB());//设置前景色

return qrConfig;//返回生成的二维码

}

}

package org.example.config;

import org.example.service.SignService;

import org.springframework.context.annotation.Configuration;

import org.springframework.web.servlet.config.annotation.CorsRegistry;

import org.springframework.web.servlet.config.annotation.InterceptorRegistry;

import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

/\*\*

\* @author zjy

\*/

@Configuration

public class WebMvcConfig implements WebMvcConfigurer {

private final SignService signService;

public WebMvcConfig(SignService signService) {

this.signService = signService;

}

@Override

public void addInterceptors(InterceptorRegistry registry) {

registry.addInterceptor(new MyInterceptor(signService)).addPathPatterns("/\*\*");

}

@Override

public void addCorsMappings(CorsRegistry registry) {

//跨域问题

registry.addMapping("/\*\*")

.allowCredentials(true)

.allowedHeaders("\*")

.allowedMethods("\*")

.allowedOrigins("\*");

}

}

package org.example.controller;

import com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;

import com.mysql.cj.util.StringUtils;

import org.example.domain.Result;

import org.example.domain.Teacher;

import org.example.service.TeacherService;

import org.springframework.web.bind.annotation.\*;

/\*\*

\* @author zjy

\*/

@RestController

//表示所有类都会返回JSON格式 spring的一个注解

@RequestMapping("/authorizations")

//请求方法是request,接口路径是/authorizations,其实是IP+端口号+路径

public class LoginController {

private final TeacherService service;

public LoginController(TeacherService service) {

this.service = service;

}

@GetMapping("/{account}/{password}")

//get方法请求参数包括账户和密码，/authorizations/{account}/{password}

public Result login(@PathVariable String account, @PathVariable String password) {

Teacher teacher = new Teacher(account, password);

//实现teacher类返回Teacher

System.out.println(teacher);

if (service.getTeacher(teacher.getAccount())==null) {

return new Result(false, "账户名错误");

}

return new Result(true, teacher);

}

@PostMapping

public Result register(@RequestBody Teacher teacher) {

Boolean flag = service.saveTeacher(teacher);

return new Result(flag, null);

}

@PostMapping("/login")

//get方法请求参数包括账户和密码，/authorizations/{account}/{password}

public Result login(@RequestBody Teacher teacher) {

if (teacher != null) {

Teacher one = service.getOne(new LambdaQueryWrapper<Teacher>()

.eq(teacher.getAccount() != null, Teacher::getAccount, teacher.getAccount())

.eq(!StringUtils.isNullOrEmpty(teacher.getPassword()), Teacher::getPassword, teacher.getPassword()));

if (one != null) {

return new Result(true, "登陆成功！");

}

}

return new Result(false,"账号或密码错误！");

}

}

package org.example.controller;

import com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;

import org.example.domain.Result;

import org.example.domain.Sign;

import org.example.domain.Student;

import org.example.service.QrService;

import org.example.service.SignService;

import org.example.service.StuClassService;

import org.example.service.StudentService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

@RestController

@RequestMapping("/manages")

public class ManageController {

@Autowired

private StudentService service;

@Autowired

private SignService signService;

@Autowired

private QrService qrService;

@Autowired

private StuClassService service2;

private List<String> className = new ArrayList<String>();

private List<String> usedClassName = new ArrayList<String>();

@GetMapping

public Result getAll() {

List<Student> list = service.list();

return new Result(true, list);

}

@GetMapping("/select")

public Result getConditionAll(String sname, String sno, String sgender, String sclass) {

System.out.println(sname + "\n" + sno + "\n" + sgender + "\n" + sclass);

LambdaQueryWrapper<Student> lqw = new LambdaQueryWrapper<>();

lqw.like(!sname.isEmpty(), Student::getSname, sname)

.like(!sno.isEmpty(), Student::getSno, sno)

.like(!sclass.isEmpty(), Student::getSclass, sclass)

.eq(!sgender.isEmpty(), Student::getSgender, sgender);

List<Student> list = service.list(lqw);

return new Result(list != null ? true : false, list);

}

@GetMapping("/create")

public Result createClass(String createclassname) {

if (usedClassName.contains(createclassname)) {

return new Result(false, null);

}

Boolean aBoolean = service2.insertClass(createclassname);

return new Result(aBoolean, null);

}

@DeleteMapping("/delete")

public Result deleteStu(String sno) {

return new Result(service.deleteStu(sno), null);

}

@GetMapping("/class")

public Result getAllClass() {

service2.list().forEach((c) -> {

usedClassName.add(c.getSclass());

});

return new Result(true, service2.list());

}

@PostMapping("/addStu")

public Result addStu(@RequestBody Student student) {

System.out.println(student);

boolean save = service.save(student);

return new Result(save, null);

}

@GetMapping("/qrcode")

public void generateV3(String content, String sclass, HttpServletResponse servletResponse) throws IOException {

className.add(sclass);

qrService.generateStream(content, servletResponse);

}

@GetMapping("/sign")

public Result sign(String sno, String sclass, HttpServletRequest request) {

if (className.contains(sclass)) {

//改sql语句

String signedIPAddress = request.getRemoteAddr();

// Boolean aBoolean = signService.updateSigned(sno);

boolean aBoolean = signService.trySign(sno,signedIPAddress);

return new Result(aBoolean, aBoolean ? "签到成功！" : "签到失败");

} else {

return new Result(false, "学号不存在或所在班级未开启签到！");

}

}

@GetMapping("/getAllsign")

public Result getAllSign() {

return new Result(true, signService.list());

}

@GetMapping("/getsign")

public Result getSign() {

if (className.contains("0000")) {

signService.deleteByclassName(className);

signService.insertSign(className);

} else {

className.add("0000");

}

return new Result(true, signService.list());

}

@GetMapping("/abortsign")

public Result abortSign(String sclass) {

if (className.contains(sclass)) {

className.remove(sclass);

return new Result(true, null);

} else {

return new Result(false, null);

}

}

@GetMapping("/signed")

public Result signed(String condition) {

LambdaQueryWrapper<Sign> lqw = new LambdaQueryWrapper<>();

lqw.eq("signed".equals(condition), Sign::getStatus, "已签到");

lqw.eq("unsigned".equals(condition), Sign::getStatus, "未签到");

return new Result(true, signService.list(lqw));

}

@GetMapping("allsignbynlassname")

public Result allSignByClassName(String sclass, String condition) {

LambdaQueryWrapper<Sign> lqw = new LambdaQueryWrapper<>();

lqw.eq(!sclass.isEmpty(), Sign::getSclass, sclass)

.eq("signed".equals(condition), Sign::getStatus, "已签到")

.eq("unsigned".equals(condition), Sign::getStatus, "未签到");

return new Result(true, signService.list(lqw));

}

}

package org.example.dao;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;

import org.apache.ibatis.annotations.\*;

import org.example.domain.Sign;

import java.util.List;

@Mapper

public interface SignDAO extends BaseMapper<Sign> {

// @Delete("DELETE FROM tb\_sign where sno in (SELECT sno FROM tb\_student where sclass in #{list})")

@Delete("<script>"

+ "DELETE FROM tb\_sign "

+ "<where>"

+ "sno in (SELECT sno FROM tb\_student "

+ "<where>"

+ "<if test='list != null'>"

+ "sclass in "

+ "<foreach item ='i' index = 'index' collection = 'list' open = '(' separator=',' close=')'>"

+ "#{i}"

+ "</foreach>"

+ "</if>"

+ "</where>"

+ ")"

+ "</where>"

+ "</script>")

void deleteByclassName(@Param("list") List<String> className);

@Insert("<script>"

+ "INSERT INTO tb\_sign(sno,sname,sgender,sclass) SELECT \* FROM tb\_student "

+ "<where>"

+ "<if test='list != null'>"

+ " sclass in"

+ "<foreach item ='i' index = 'index' collection = 'list' open = '(' separator=',' close=')'>"

+ "#{i}"

+ "</foreach>"

+ "</if>"

+ "</where>"

+ "</script>")

void insertSign(@Param("list") List<String> className);

@Update("UPDATE tb\_sign SET status = '已签到' where sno = #{sno}")

Boolean updateSigned(String sno);

}

package org.example.dao;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;

import org.apache.ibatis.annotations.Insert;

import org.apache.ibatis.annotations.Mapper;

import org.example.domain.StuClass;

@Mapper

public interface StuClassDAO extends BaseMapper<StuClass> {

@Insert("INSERT INTO tb\_class VALUES (null,#{sclass})")

Boolean insertClass(String sclass);

}

package org.example.dao;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;

import org.apache.ibatis.annotations.Delete;

import org.apache.ibatis.annotations.Mapper;

import org.example.domain.Student;

@Mapper

public interface StudentDAO extends BaseMapper<Student> {

@Delete("DELETE FROM tb\_student where sno=#{sno}")

Boolean deleteBySno(String sno);

}

package org.example.dao;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;

import org.apache.ibatis.annotations.Insert;

import org.apache.ibatis.annotations.Mapper;

import org.apache.ibatis.annotations.Select;

import org.example.domain.Teacher;

@Mapper

public interface TeacherDAO extends BaseMapper<Teacher> {

@Select("SELECT \* FROM tb\_teacher where account = #{account}")

Teacher getByAccount(String account);

//查该账户后是否存在

@Select("SELECT \* FROM tb\_teacher where account = #{account} and password = #{password}")

Boolean result(Teacher teacher);

@Insert("INSERT INTO tb\_teacher VALUES(#{account},#{password})")

Boolean save(Teacher teacher);

}

package org.example.domain;

import lombok.Data;

@Data

public class Result {

private boolean flag;

private Object data;

public Result(boolean flag, Object data) {

this.flag = flag;

this.data = data;

}

}

package org.example.domain;

import com.baomidou.mybatisplus.annotation.TableName;

import lombok.Data;

@Data

@TableName("tb\_sign")

public class Sign {

private String sno;

private String sname;

private String sgender;

private String sclass;

private String status;

private String ip;

}

package org.example.domain;

import com.baomidou.mybatisplus.annotation.TableName;

import lombok.Data;

@Data

@TableName("tb\_class")

public class StuClass {

private String sclass;

private Integer id;

}

package org.example.domain;

import com.baomidou.mybatisplus.annotation.TableName;

import lombok.Data;

@Data

@TableName("tb\_student")

public class Student {

private String sno;

private String sname;

private String sgender;

private String sclass;

}

package org.example.domain;

import com.baomidou.mybatisplus.annotation.TableName;

import lombok.Data;

@Data

@TableName("tb\_teacher")

public class Teacher {

private String account;

private String password;

public Teacher(String account, String password) {

this.account = account;

this.password = password;

}

public Teacher() {

}

}

package org.example.service;

import javax.servlet.http.HttpServletResponse;

import java.io.File;

public interface QrService {

void generateFile(String content, File file);

void generateStream(String content, HttpServletResponse response);

}

package org.example.service.impl;

import cn.hutool.extra.qrcode.QrCodeUtil;

import cn.hutool.extra.qrcode.QrConfig;

import org.example.service.QrService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import javax.servlet.http.HttpServletResponse;

import java.io.File;

import java.io.IOException;

@Service

public class QrServiceImpl implements QrService {

@Autowired

QrConfig qrConfig;

@Override

public void generateFile(String content, File file) {

QrCodeUtil.generate(content,qrConfig,file);

}

@Override

public void generateStream(String content, HttpServletResponse response) {

try {

QrCodeUtil.generate(content,qrConfig,"png",response.getOutputStream());

} catch (IOException e) {

System.out.println("this");

System.out.println(e.getMessage());

throw new RuntimeException(e);

}

}

}

package org.example.service.impl;

import com.baomidou.mybatisplus.core.conditions.update.LambdaUpdateWrapper;

import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;

import org.example.dao.SignDAO;

import org.example.domain.Sign;

import org.example.service.SignService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class SignServiceImpl extends ServiceImpl<SignDAO, Sign> implements SignService {

@Autowired

private SignDAO dao;

@Override

public void deleteByclassName(List<String> className) {

dao.deleteByclassName(className);

}

@Override

public void insertSign(List<String> className) {

dao.insertSign(className);

}

@Override

public Boolean updateSigned(String sno) {

return dao.updateSigned(sno);

}

@Override

public boolean trySign(String sno, String signedIPAddress) {

Sign sign = new Sign();

sign.setStatus("已签到");

sign.setIp(signedIPAddress);

return update(sign,new LambdaUpdateWrapper<Sign>().eq(Sign::getSno,sno));

}

}

package org.example.service.impl;

import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;

import org.example.dao.StuClassDAO;

import org.example.domain.StuClass;

import org.example.service.StuClassService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

@Service

public class StuClassServiceImpl extends ServiceImpl<StuClassDAO, StuClass> implements StuClassService {

@Autowired

private StuClassDAO dao;

@Override

public Boolean insertClass(String sclass) {

Boolean aBoolean = dao.insertClass(sclass);

return aBoolean;

}

}

package org.example.service.impl;

import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;

import org.example.dao.StudentDAO;

import org.example.domain.Student;

import org.example.service.StudentService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

@Service

public class StudentServiceimpl extends ServiceImpl<StudentDAO, Student> implements StudentService {

@Autowired

private StudentDAO dao;

@Override

public Boolean deleteStu(String sno) {

return dao.deleteBySno(sno);

}

}

package org.example.service.impl;

import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;

import org.example.dao.TeacherDAO;

import org.example.domain.Teacher;

import org.example.service.TeacherService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

@Service

public class TeacherServiceImpl extends ServiceImpl<TeacherDAO, Teacher> implements TeacherService {

@Autowired

private TeacherDAO dao;

@Override

public Teacher getTeacher(String account) {

return dao.getByAccount(account);

}

@Override

public Boolean saveTeacher(Teacher teacher) {

Teacher teacher1 = getTeacher(teacher.getAccount());

if(teacher1 == null) {

Boolean save = dao.save(teacher);

return save;

}

else{

return false;

}

}

}

package org.example.service;

import com.baomidou.mybatisplus.extension.service.IService;

import org.example.domain.Sign;

import java.util.List;

public interface SignService extends IService<Sign> {

void deleteByclassName(List<String> className);

void insertSign(List<String> className);

Boolean updateSigned(String sno);

boolean trySign(String sno, String signedIPAddress);

}

package org.example.service;

import com.baomidou.mybatisplus.extension.service.IService;

import org.example.domain.StuClass;

public interface StuClassService extends IService<StuClass> {

Boolean insertClass(String sclass);

}

package org.example.service;

import com.baomidou.mybatisplus.extension.service.IService;

import org.example.domain.Student;

public interface StudentService extends IService<Student> {

Boolean deleteStu(String sno);

}

package org.example.service;

import com.baomidou.mybatisplus.extension.service.IService;

import org.example.domain.Teacher;

public interface TeacherService extends IService<Teacher>{

Teacher getTeacher(String account);

Boolean saveTeacher(Teacher teacher);

}

package org.example;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.web.servlet.support.SpringBootServletInitializer;

/\*\*

\* @author zjy

\*/

@SpringBootApplication

public class SignApplication extends SpringBootServletInitializer {

public static void main(String[] args) {

SpringApplication.run(SignApplication.class, args);

}

}

请完整的分析该文档的代码，给出每个类的解释，并画出流程图和UML类图