package fun.redamancyxun.chinese.backend;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.scheduling.annotation.EnableScheduling;

@SpringBootApplication

@EnableScheduling

public class ChineseScoringModelApplication {

public static void main(String[] args) {

SpringApplication.run(ChineseScoringModelApplication.class, args);

}

}

package fun.redamancyxun.chinese.backend.common;

public class CommonConstants {

public final static String SESSION = "session";

public final static String USER\_FILE\_PATH = "/var/www/html/images/";

public final static String AUDIO\_FILE\_PATH = "/audio/";

public final static String WINDOWS\_SCORE\_FILE\_PATH = "src/main/java/fun/redamancyxun/chinese/backend/score/";

public final static String LINUX\_SCORE\_FILE\_PATH = "/score/";

// public final static String LINUX\_SCORE\_FILE\_PATH = "src/main/java/fun/redamancyxun/chinese/backend/score/";

public final static String WINDOWS\_PYTHON\_PATH = "D:\\用户\\桌面\\国际中文朗读语料评分模型建构与优化实践\\模型训练\\训练工具\\训练数据处理\\.venv\\Scripts\\python.exe";

public final static String LINUX\_PYTHON\_PATH = "/python/venv/bin/python3";

// public final static String LINUX\_PYTHON\_PATH = "D:\\用户\\桌面\\国际中文朗读语料评分模型建构与优化实践\\模型训练\\训练工具\\训练数据处理\\.venv\\Scripts\\python.exe";

public final static String WINDOWS\_RESOURCES\_STATIC\_PATH = "src/main/resources/static/";

public final static String LINUX\_RESOURCES\_STATIC\_PATH = "/resources/";

// public final static String LINUX\_RESOURCES\_STATIC\_PATH = "src/main/resources/static/";

public final static String VERIFICATION\_TITLE = "验证码";

public final static String AUTH\_FILE\_PATH = "/home/ubuntu/ver/";

public final static String IMAGE\_PATH = "http://chinese.redamancyxun.fun/images/";

public final static String AUTH\_DOWN\_PATH = "https://redamancyxun.fun/image2/";

public final static String SEPARATOR = ",";

public final static String CHAT\_RECORD\_COLLECTION\_NAME = "chat\_record";

public final static String WX\_SESSION\_REQUEST\_URL = "https://api.weixin.qq.com/sns/jscode2session";

public final static String SHADOW\_TEST = "shadow";

//https://pay.weixin.qq.com/wiki/doc/apiv3/apis/chapter3\_5\_1.shtml

public final static String VIEW = "VIEW\_COUNT";

public final static String WX\_PAY\_REQUEST\_URL = "https://api.mch.weixin.qq.com/v3/pay/transactions/jsapi";

public final static String CNY\_CURRENCY = "CNY";

public final static String SIGN\_TYPE\_RSA = "RSA";

public final static String SIGN\_TYPE\_HMAC\_SHA256 = "HMAC-SHA256";

public final static String LANG\_TYPE\_ZH\_CN = "zh\_CN";

public final static String DEFAULT\_SCHOOL = "华东师范大学";

public static final String DOWNLOAD\_PATH = "";

}

package fun.redamancyxun.chinese.backend.common;

import lombok.Getter;

import java.util.HashMap;

import java.util.Map;

/\*\*

\* @author xiaowen

\* @version 2022/1/19 19:21

\*/

@Getter

public enum CommonErrorCode {

//1打头是微信错误，其他是程序错误

WX\_LOGIN\_BUSY(1002,"系统繁忙，此时请开发者稍候再试","微信小程序繁忙，请稍候再试"),

WX\_LOGIN\_INVALID\_CODE(1003,"无效的code","授权失败，请检查微信账号是否正常"),

WX\_LOGIN\_FREQUENCY\_REFUSED(1004,"请求太频繁，一分钟不能超过100次","请勿多次重复授权"),

WX\_LOGIN\_UNKNOWN\_ERROR(1005,"微信授权未知错误","微信异常，请稍后再试"),

WX\_APPSECRET\_INVALID(1006,"AppSecret 错误或者 AppSecret 不属于这个小程序","系统异常，请稍后再试"),

WX\_GRANTTYPE\_MUSTBE\_CLIENTCREDENTIAL(1007,"请确保 grant\_type 字段值为 client\_credential","系统异常，请稍后再试"),

WX\_APPID\_INVALID(1008,"不合法的 AppID","系统异常，请稍后再试"),

WX\_CONTENT\_ILLEGAL(1009,"内容安全校验不通过","内容含有违法违规内容，请修改"),

WX\_CONTENT\_SECURITY\_FORMAT\_ERROR(1010,"内容安全校验格式不合法","系统异常，请稍后再试"),

//微信支付回调

WX\_NOTIFY\_RESULT\_NULL(1011,"回调结果为空","回调结果为空"),

WX\_NOTIFY\_RESULT\_UNEXPECTED(1011,"回调结果不是success","回调结果不是success"),

//微信订阅消息

WX\_SUBSCRIBE\_SEND\_NULL(140000,"订阅消息返回体为空","系统异常，请稍后再试"),

WX\_SUBSCRIBE\_SEND\_40003(140003,"touser字段openid为空或者不正确","系统异常，请稍后再试"),

WX\_SUBSCRIBE\_SEND\_40037(140037,"订阅模板id为空或不正确","系统异常，请稍后再试"),

WX\_SUBSCRIBE\_SEND\_43101(143101,"用户拒绝接受消息，如果用户之前曾经订阅过，则表示用户取消了订阅关系","系统异常，请稍后再试"),

WX\_SUBSCRIBE\_SEND\_47003(147003,"模板参数不准确，可能为空或者不满足规则，errmsg会提示具体是哪个字段出错","系统异常，请稍后再试"),

WX\_SUBSCRIBE\_SEND\_41030(141030,"page路径不正确，需要保证在现网版本小程序中存在，与app.json保持一致","系统异常，请稍后再试"),

//微信退款

//微信二维码

WX\_QRCODE\_UNAUTHORIZED(1012,"暂无生成权限","系统异常，请稍后再试"),

WX\_QRCODE\_TOO\_FREQUENT(1013,"调用分钟频率受限(目前5000次/分钟，会调整)，如需大量小程序码，建议预生成","系统繁忙，请稍后重试"),

USER\_NOT\_EXIST(2001,"用户不存在","用户不存在"),

SYSTEM\_ERROR(2002,"系统错误","系统错误，请重试"),

INVALID\_SESSION(2006,"会话丢失","登录已失效，请重新登录"),

SCHOOL\_UNAUTHORIZED(2007,"未通过学校认证","尚未进行校园认证，请先认证"),

INVALID\_PICTURE\_TYPE(2008,"无效的图片类型（必须是goods或advice）","图片上传出错，请重试"),

TEL\_USED\_ERROR(2009,"手机号已注册","请前往登录"),

VERIFY\_FAILED(2010,"验证失败","请重试"),

LOGIN\_FAILED(2011,"登录失败","用户名或密码错误"),

PARAMS\_INVALID(2012,"存在有误的参数","请重试"),

UNSIGNED\_USER(2013,"未注册用户","请前往注册"),

INVALID\_PHONE(2014,"无效手机号","请输入正确的手机号"),

UPDATE\_FAIL(2015,"更新失败，出现竞态条件","请稍后重试"),

USER\_NOT\_ADMIN(2016,"用户非管理员","用户非管理员"),

NEED\_SESSION\_ID(2017,"未传入sessionId","请传入会话id"),

LOGIN\_HAS\_OVERDUE(2018,"登录已过期","登录已过期"),

SESSION\_IS\_INVALID(2019,"该session数据库里没有","请在header里key为session处对应有效的sessionId"),

DUPLICATE\_DATABASE\_INFORMATION(2020,"重复的数据库信息","信息添加失败，请重试"),

UPLOAD\_FILE\_FAIL(2021,"上传文件失败","请检查网络状况后稍后重试"),

FILENAME\_CAN\_NOT\_BE\_NULL(2022,"文件名不能为空","请取一个有后缀的文件名"),

EXECUTION\_FAIL(2023,"方法执行错误","请稍后重试"),

DATA\_NOT\_EXISTS(2024, "无效的数据库信息", "数据库查询失败"),

INVALID\_PARAM(2025, "非法参数", "参数非法，请检查输入"),

OPERATION\_TOO\_FREQUENT(2026, "操作过于频繁", "操作过于频繁，请稍后再试"),

USERNAME\_HAS\_BEEN\_SIGNED\_UP(2027,"用户名已被注册","请尝试新的用户名"),

PASSWORD\_NOT\_RIGHT(2028,"密码错误","密码错误，请重新输入密码"),

TWO\_INPUTED\_PASSWORDS\_NOT\_THE\_SAME(2028,"两次输入的密码不一致","两次输入密码不一致，请重新输入"),

READ\_FILE\_ERROR(2029,"读取文件失败","请检查文件格式后重新上传文件"),

FILE\_NOT\_EXIST(2030,"文件不存在","文件不存在"),

DOWNLOAD\_FILE\_FAILED(2031,"下载失败","请在浏览器地址栏中输入链接来测试，或者检查网络或系统状况"),

CAN\_NOT\_MODIFY(2033, "不可修改", "无权修改"),

ACCOUNT\_STATUS\_NORMAL(2034, "账号状态正常","账号状态正常，无需申诉"),

HAVE\_NO\_PERMISSION(2035, "无权进行本次操作","无权进行本次操作"),

TELEPHONE\_HAS\_BEEN\_SIGNED\_UP(2036,"手机号已被注册","手机号已被注册"),

PASSWORD\_NOT\_QUANTIFIED(2037,"密码不符合要求","密码不符合要求"),

ZIPPATH\_ERROR(2039,"打包路径错误，必须以.zip结尾","打包路径错误，必须以.zip结尾"),

MESSAGE\_NOT\_EXIST(2040,"消息不存在","消息不存在"),

ARTICLE\_NOT\_FOUND(2041,"文章不存在","文章不存在"),

ARTICLE\_ALREADY\_EXIST(2042,"文章已存在","文章已存在"),

CARD\_NOT\_FOUND(2043,"卡片不存在","卡片不存在"),

VIDEO\_ALREADY\_EXIST(2044,"视频已存在","视频已存在"),

THEME\_NOT\_EXIST(2045,"主题不存在","主题不存在"),

ROLE\_ALREADY\_SET(2046,"角色已设置","角色已设置"),

SUBPROJECT\_NOT\_EXIST(2047,"子项目不存在","子项目不存在"),

GUIDANCE\_NOT\_EXIST(2048,"指导不存在","指导不存在"),

FEEDBACK\_NOT\_EXIST(2049,"反馈不存在","反馈不存在"),

FEEDBACK\_HAS\_BEEN\_DEALED(2050,"反馈已处理","反馈已处理"),

OBSERVATION\_NOT\_EXIST(2051,"观察不存在","观察不存在"),

QUESTION\_NOT\_EXIST(2052,"问题不存在","问题不存在"),

QUESTION\_ALREADY\_DELETED(2053,"问题已删除","问题已删除"),

QUESTIONNAIRE\_NOT\_EXIST(2054,"问卷不存在","问卷不存在"),

QUESTION\_RESULT\_NOT\_EXIST(2055,"问题结果不存在","问题结果不存在"),

SUB\_PROJECT\_NOT\_EXIST(2056,"子项目不存在","子项目不存在"),

QUESTIONNAIRE\_FINISHED(2057,"问卷已完成","问卷已完成"),

USER\_ROLE\_NOT\_SET(2058,"用户身份未设置","用户身份未设置"),

DELETE\_ERROR(2059,"删除失败","删除失败,检查参数是否正确"),

QUESTION\_SECTION\_NOT\_EXIST(2060,"问题板块不存在","问题板块不存在"),

;

/\*\*

\* 错误码

\*/

private final Integer errorCode;

/\*\*

\* 错误原因（给开发看的）

\*/

private final String errorReason;

/\*\*

\* 错误行动指示（给用户看的）

\*/

private final String errorSuggestion;

CommonErrorCode(Integer errorCode, String errorReason, String errorSuggestion) {

this.errorCode = errorCode;

this.errorReason = errorReason;

this.errorSuggestion = errorSuggestion;

}

@Override

public String toString() {

return "CommonErrorCode{" +

"errorCode=" + errorCode +

", errorReason='" + errorReason + '\'' +

", errorSuggestion='" + errorSuggestion + '\'' +

'}';

}

//use for json serialization

public Map<String,Object> toMap(){

Map<String,Object> map = new HashMap<>();

map.put("errorCode",errorCode);

map.put("errorReason",errorReason);

map.put("errorSuggestion",errorSuggestion);

return map;

}

}

package fun.redamancyxun.chinese.backend.common;

import com.github.pagehelper.PageInfo;

import lombok.AllArgsConstructor;

import lombok.Builder;

import lombok.Data;

import lombok.NoArgsConstructor;

import java.util.List;

@Data

@NoArgsConstructor

@AllArgsConstructor

@Builder

public class Page<T> {

//当前页

@Builder.Default

private Integer pageNum = 1;

//每页显示总条数

@Builder.Default

private Integer pageSize = 10;

//总条数

private Long total;

//总页数

private Integer pages;

//分页结果

private List<T> items;

public Page(PageInfo<T> pageInfo) {

this.pageNum = pageInfo.getPageNum();

this.pageSize = pageInfo.getPageSize();

this.total = pageInfo.getTotal();

this.pages = pageInfo.getPages();

this.items = pageInfo.getList();

}

public Page(PageParam pageParam,Long total,Integer pages,List<T> list){

this.pageNum = pageParam.getPageNum();

this.pageSize = pageParam.getPageSize();

this.total = total;

this.pages = pages;

this.items = list;

}

}

package fun.redamancyxun.chinese.backend.common;

import io.swagger.annotations.ApiModel;

import io.swagger.annotations.ApiModelProperty;

import lombok.AllArgsConstructor;

import lombok.Builder;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@NoArgsConstructor

@AllArgsConstructor

@Builder

@ApiModel

public class PageParam {

@ApiModelProperty("每页显示数量 (不小于0)")

private Integer pageSize=10;

@ApiModelProperty("页数 (不小于0)")

private Integer pageNum=0;

@ApiModelProperty("格式:字段名 排序规律,例:\"id asc\"")

private String orderBy="id asc";

}

package fun.redamancyxun.chinese.backend.common;

import fun.redamancyxun.chinese.backend.exception.EnumExceptionType;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@Data

@NoArgsConstructor

@AllArgsConstructor

public class Result {

private int code;

private String message;

private Object result;

public static Result success(Object data) {

return success("操作成功", data);

}

public static Result success(String mess, Object data) {

Result m = new Result();

m.setCode(0);

m.setResult(data);

m.setMessage(mess);

return m;

}

public static Result fail(String mess) {

return fail(mess, null);

}

public static Result fail(String mess, Object data) {

Result m = new Result();

m.setCode(-1);

m.setResult(data);

m.setMessage(mess);

return m;

}

public static Result result(EnumExceptionType enumExceptionType, Object data) {

Result m = new Result();

m.setCode(enumExceptionType.getErrorCode());

m.setResult(data);

m.setMessage(enumExceptionType.getCodeMessage());

return m;

}

public static Result result(EnumExceptionType enumExceptionType) {

return result(enumExceptionType,null);

}

}

package fun.redamancyxun.chinese.backend.config;

import org.springframework.context.annotation.Configuration;

import org.springframework.scheduling.annotation.AsyncConfigurer;

import org.springframework.scheduling.annotation.EnableAsync;

import org.springframework.scheduling.concurrent.ThreadPoolTaskExecutor;

import java.util.concurrent.Executor;

@Configuration

@EnableAsync

public class AsyncConfig implements AsyncConfigurer {

@Override

public Executor getAsyncExecutor() {

ThreadPoolTaskExecutor executor = new ThreadPoolTaskExecutor();

executor.setCorePoolSize(2);

executor.setMaxPoolSize(5);

executor.setQueueCapacity(100);

executor.setThreadNamePrefix("Async-");

executor.initialize();

return executor;

}

}

package fun.redamancyxun.chinese.backend.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.web.cors.CorsConfiguration;

import org.springframework.web.cors.UrlBasedCorsConfigurationSource;

import org.springframework.web.filter.CorsFilter;

import java.util.Collections;

@Configuration

public class CorsConfig {

@Bean

public CorsFilter corsFilter() {

CorsConfiguration corsConfiguration = new CorsConfiguration();

//1.允许任何来源

corsConfiguration.setAllowedOrigins(Collections.singletonList("\*"));

//2.允许任何请求头

corsConfiguration.addAllowedHeader(CorsConfiguration.ALL);

//3.允许任何方法

corsConfiguration.addAllowedMethod(CorsConfiguration.ALL);

//4.允许凭证

corsConfiguration.setAllowCredentials(true);

UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();

source.registerCorsConfiguration("/\*\*", corsConfiguration);

return new CorsFilter(source);

}

}

package fun.redamancyxun.chinese.backend.config;

import com.google.gson.Gson;

import com.google.gson.GsonBuilder;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

@Configuration

public class JsonConfig {

//将Bean命名为json,调用时创建一个Gson对象

@Bean("json")

public Gson json(){

return new GsonBuilder().serializeNulls().create();

}

}

package fun.redamancyxun.chinese.backend.config;

import org.springframework.boot.web.servlet.MultipartConfigFactory;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.util.unit.DataSize;

import org.springframework.util.unit.DataUnit;

import javax.servlet.MultipartConfigElement;

@Configuration

public class MultipartConfig {

//设置上传文件大小限制

@Bean

public MultipartConfigElement multipartConfigElement() {

MultipartConfigFactory config = new MultipartConfigFactory();

config.setMaxFileSize(DataSize.of(10, DataUnit.MEGABYTES));

config.setMaxRequestSize(DataSize.of(10, DataUnit.MEGABYTES));

return config.createMultipartConfig();

}

}

package fun.redamancyxun.chinese.backend.config;

import fun.redamancyxun.chinese.backend.util.SpringInterceptor;

import org.springframework.context.annotation.Configuration;

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

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

@Configuration

class MvcConfig implements WebMvcConfigurer {

@Override

public void addInterceptors(InterceptorRegistry registry) {

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

}

}

package fun.redamancyxun.chinese.backend.config;

import com.baomidou.mybatisplus.core.handlers.MetaObjectHandler;

import lombok.extern.slf4j.Slf4j;

import org.apache.ibatis.reflection.MetaObject;

import org.springframework.stereotype.Component;

import java.time.LocalDateTime;

@Slf4j

@Component

public class MyMetaObjectHandler implements MetaObjectHandler {

@Override

public void insertFill(MetaObject metaObject) {

if (this.getFieldValByName("createTime",metaObject) == null)

this.setFieldValByName("createTime", LocalDateTime.now(),metaObject);

}

@Override

public void updateFill(MetaObject metaObject) {

if (this.getFieldValByName("updateTime",metaObject) == null)

this.setFieldValByName("updateTime", LocalDateTime.now(),metaObject);

}

}

package fun.redamancyxun.chinese.backend.config;

import com.baomidou.mybatisplus.extension.plugins.OptimisticLockerInterceptor;

import com.baomidou.mybatisplus.extension.plugins.PaginationInterceptor;

import com.baomidou.mybatisplus.extension.spring.MybatisSqlSessionFactoryBean;

import org.apache.ibatis.session.SqlSessionFactory;

import org.mybatis.spring.SqlSessionTemplate;

import org.mybatis.spring.annotation.MapperScan;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.transaction.annotation.EnableTransactionManagement;

import javax.sql.DataSource;

@EnableTransactionManagement

@Configuration

@MapperScan("fun.redamancyxun.chinese.backend.mapper")

public class MybatisPlusConfig {

/\*\*

\* mybatisPlus插件

\*/

@Bean

public OptimisticLockerInterceptor optimisticLockerInterceptor() {

return new OptimisticLockerInterceptor();

}

@Bean

public PaginationInterceptor paginationInterceptor(){

return new PaginationInterceptor();

}

// @Bean

// public SqlSessionFactory sqlSessionFactory(DataSource dataSource) throws Exception {

// MybatisSqlSessionFactoryBean factoryBean = new MybatisSqlSessionFactoryBean();

// factoryBean.setDataSource(dataSource);

// return factoryBean.getObject();

// }

//

// @Bean

// public SqlSessionTemplate sqlSessionTemplate(SqlSessionFactory sqlSessionFactory) {

// return new SqlSessionTemplate(sqlSessionFactory);

// }

}

package fun.redamancyxun.chinese.backend.config;

import org.springframework.amqp.core.Queue;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

@Configuration

public class RabbitConfig {

@Bean

public Queue queue() {

return new Queue("q\_hello");

}

}

package fun.redamancyxun.chinese.backend.config;

import com.alibaba.fastjson.support.spring.GenericFastJsonRedisSerializer;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.data.redis.connection.RedisConnectionFactory;

import org.springframework.data.redis.core.RedisTemplate;

import org.springframework.data.redis.serializer.RedisSerializer;

import org.springframework.data.redis.serializer.StringRedisSerializer;

/\*\*

\* 重写RedisTemplate的序列化方式

\*/

@Configuration

public class RedisConfig {

@Bean

public RedisSerializer<String> redisKeySerializer() {

return new StringRedisSerializer();

}

@Bean

public RedisSerializer<Object> redisValueSerializer() {

return new GenericFastJsonRedisSerializer();

}

/\*\*

\* RedisTemplate配置

\*

\* @param factory

\*/

@Bean

public RedisTemplate<String, Object> redisTemplate(RedisConnectionFactory factory, RedisSerializer<String> redisKeySerializer, RedisSerializer<Object> redisValueSerializer) {

RedisTemplate<String, Object> redisTemplate = new RedisTemplate<>();

redisTemplate.setConnectionFactory(factory);

//设置Key的序列化采用StringRedisSerializer

redisTemplate.setKeySerializer(redisKeySerializer);

redisTemplate.setHashKeySerializer(redisKeySerializer);

//设置值的序列化

redisTemplate.setValueSerializer(redisValueSerializer);

redisTemplate.setHashValueSerializer(redisValueSerializer);

redisTemplate.afterPropertiesSet();

return redisTemplate;

}

}

package fun.redamancyxun.chinese.backend.config;

import lombok.Setter;

import org.springframework.boot.context.properties.ConfigurationProperties;

import org.springframework.context.annotation.Configuration;

import java.util.HashMap;

import java.util.Map;

import java.util.Optional;

/\*\*

\* 服务熔断配置

\*

\* @author miracle

\* @since 2021/11/22 17:57

\*/

@Setter

@Configuration

@ConfigurationProperties("server-fuse")

public class ServerFuseConfig {

/\*\* 服务熔断（限流）总开关 \*/

private Boolean fuse = Boolean.FALSE;

/\*\* 并发达到峰值时，是否允许继续执行程序，默认值false \*/

private Boolean process = Boolean.FALSE;

/\*\* 接口并发数量 \*/

private Integer concurrency = 10;

/\*\* 自定义接口并发数配置 \*/

private Map<String, Integer> customConcurrencyMap = new HashMap<>(16);

public Boolean getFuse() {

return Optional.ofNullable(fuse).orElse(Boolean.FALSE);

}

public Boolean getProcess() {

return Optional.ofNullable(process).orElse(Boolean.FALSE);

}

public Integer getConcurrency() {

return Optional.ofNullable(concurrency).orElse(10);

}

public Map<String, Integer> getCustomConcurrencyMap() {

return Optional.ofNullable(customConcurrencyMap).orElseGet(HashMap::new);

}

}

//package fun.redamancyxun.chinese.backend.config;

//

//import org.springframework.context.annotation.Bean;

//import org.springframework.context.annotation.Configuration;

//import springfox.documentation.builders.ApiInfoBuilder;

//import springfox.documentation.oas.annotations.EnableOpenApi;

//import springfox.documentation.service.\*;

//import springfox.documentation.spi.DocumentationType;

//import springfox.documentation.spi.service.contexts.SecurityContext;

//import springfox.documentation.spring.web.plugins.Docket;

//

//import java.util.ArrayList;

//import java.util.Collections;

//import java.util.List;

//

//@Configuration

//@EnableOpenApi

//public class Swagger3Config {

//

// @Bean

// public Docket docket() {

// return new Docket(DocumentationType.OAS\_30).apiInfo(apiInfo())

// .securitySchemes(securitySchemes())

// .securityContexts(Collections.singletonList(securityContext()));

// }

//

// private ApiInfo apiInfo() {

// return new ApiInfoBuilder()

// .title("漫游城——citywalker社群平台先行者")

// .description("接口文档")

// //.termsOfServiceUrl("http://localhost:8080/swagger-ui.html")//数据源

// .version("1.0")

// .build();

// }

// private List<SecurityScheme> securitySchemes() {

// List<SecurityScheme> apiKeyList= new ArrayList<>();

// //注意，这里应对应登录token鉴权对应的k-v

// apiKeyList.add(new ApiKey("session", "session", "header"));

// return apiKeyList;

// }

//

// /\*\*

// \* 这里设置 swagger2 认证的安全上下文

// \*/

// private SecurityContext securityContext() {

// return SecurityContext.builder()

// .securityReferences(Collections.singletonList(new SecurityReference("session", scopes())))

// .build();

// }

//

// /\*\*

// \* 这里是写允许认证的scope

// \*/

// private AuthorizationScope[] scopes() {

// return new AuthorizationScope[]{

// new AuthorizationScope("web", "All scope is trusted!")

// };

// }

//}

package fun.redamancyxun.chinese.backend.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.web.multipart.commons.CommonsMultipartResolver;

@Configuration

public class WebConfig {

@Bean(name = "multipartResolver")

public CommonsMultipartResolver multipartResolver() {

CommonsMultipartResolver multipartResolver = new CommonsMultipartResolver();

multipartResolver.setMaxUploadSize(10485760); // 10MB

return multipartResolver;

}

}

package fun.redamancyxun.chinese.backend.config;

import org.springframework.boot.autoconfigure.ImportAutoConfiguration;

import org.springframework.context.annotation.Bean;

import org.springframework.web.socket.server.standard.ServerEndpointExporter;

@ImportAutoConfiguration//尝试在测试类中排除 WebSocket 相关的配置。使用 @ImportAutoConfiguration 注解来排除特定的自动配置类或者属性文件，从而避免加载 WebSocket 相关的配置。

//@Configuration

public class WebSocketConfig {

/\*\*

\* 注入ServerEndpointExporter，

\* 这个bean会自动注册使用了@ServerEndpoint注解声明的Websocket endpoint

\*/

@Bean

public ServerEndpointExporter serverEndpointExporter() {

return new ServerEndpointExporter();

}

}

package fun.redamancyxun.chinese.backend.config;

import lombok.Data;

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

import org.springframework.stereotype.Component;

/\*\*

\* @author yannis

\* @version 2021/1/22 22:46

\*/

@Component

@Data

public class YmlConfig {

// @Value("${server.domain}")

// private String domain;

@Value("${server.port}")

private String port;

// @Value("${uniapp.appid}")

// private String appId;

//

// @Value("${uniapp.appsecret}")

// private String appSecret;

// @Value("${mini-app.mch-id}")

// private String mchId;

//

// @Value("${mini-app.mch-serial-no}")

// private String mchSerialNo;

//

// @Value("${mini-app.mch-private-key}")

// private String mchPrivateKey;

//

// @Value("${mini-app.apiV3-key}")

// private String apiV3Key;

}

//package fun.redamancyxun.chinese.backend.controller;

//

//import fun.redamancyxun.chinese.backend.common.Result;

////import fun.redamancyxun.chinese.backend.controller.forum.request.ContactRequest;

//import fun.redamancyxun.chinese.backend.exception.MyException;

//import fun.redamancyxun.chinese.backend.service.GlobalService;

//import io.swagger.annotations.Api;

//import io.swagger.annotations.ApiImplicitParam;

//import io.swagger.annotations.ApiImplicitParams;

//import io.swagger.annotations.ApiOperation;

//import lombok.extern.slf4j.Slf4j;

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

//import org.springframework.validation.annotation.Validated;

//import org.springframework.web.bind.annotation.PostMapping;

//import org.springframework.web.bind.annotation.RequestBody;

//import org.springframework.web.bind.annotation.RestController;

//import org.springframework.web.multipart.MultipartFile;

//

//import java.util.ArrayList;

//import java.util.List;

//

//@RestController

//@Slf4j

//@Validated

//@Api("GlobalController")

//public class GlobalController {

//

// @Autowired

// private GlobalService globalService;

//

// /\*\*

// \* 上传图片

// \* @param file

// \* @return

// \*/

// @PostMapping(value = "/uploadImage", produces = "application/json")

// @ApiOperation(value = "上传图片 file:图片文件")

// @ApiImplicitParams({

// @ApiImplicitParam(name = "file", value = "图片文件", required = true, paramType = "query", dataType = "MultipartFile")})

// public Result uploadImage(@RequestBody MultipartFile file) {

// try {

// return Result.success(globalService.uploadImage(file));

// } catch (Exception e) {

// if (e instanceof MyException) {

// return Result.result(((MyException) e).getEnumExceptionType());

// }

// return Result.fail(e.getMessage());

// }

// }

//

// /\*\*

// \* 上传多个图片

// \* @param files

// \* @return

// \*/

// @PostMapping(value = "/uploadImages", produces = "application/json")

// @ApiOperation(value = "上传多个图片 files:图片文件数组")

// @ApiImplicitParams({

// @ApiImplicitParam(name = "files", value = "图片文件数组", required = true, paramType = "query", dataType = "List<MultipartFile>")})

// public Result uploadImages(@RequestBody MultipartFile[] files) {

// List<String> imageList = new ArrayList<>();

// try {

// for (MultipartFile file : files){

// imageList.add(globalService.uploadImage(file));

// }

// return Result.success(imageList);

// } catch (Exception e) {

// if (e instanceof MyException) {

// return Result.result(((MyException) e).getEnumExceptionType());

// }

// return Result.fail(e.getMessage());

// }

// }

//

//// /\*\*

//// \* 联系我们（发送邮件给客服）

//// \* @param contactRequest

//// \* @return

//// \*/

//// @Auth

//// @PostMapping(value = "/contact" , produces = "application/json")

//// @ApiOperation(value = "联系我们（发送邮件给客服）")

//// public Result contact(@RequestBody ContactRequest contactRequest){

//// try{

//// return Result.success(globalService.contact(contactRequest));

//// }catch (Exception e){

//// if(e instanceof MyException){

//// return Result.result(((MyException) e).getEnumExceptionType());

//// }

//// return Result.fail(e.getMessage());

//// }

//// }

//

//}

package fun.redamancyxun.chinese.backend.controller;

public class Test {

public static void main(String[] args) {

}

}

package fun.redamancyxun.chinese.backend.controller.lesson;

import fun.redamancyxun.chinese.backend.common.Result;

import fun.redamancyxun.chinese.backend.controller.lesson.response.LessonListResponse;

import fun.redamancyxun.chinese.backend.controller.lesson.response.LessonDetailResponse;

import fun.redamancyxun.chinese.backend.exception.MyException;

import fun.redamancyxun.chinese.backend.service.LessonService;

import fun.redamancyxun.chinese.backend.service.ScoreActionService;

import fun.redamancyxun.chinese.backend.util.SessionUtils;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiImplicitParam;

import io.swagger.annotations.ApiImplicitParams;

import io.swagger.annotations.ApiOperation;

import lombok.extern.slf4j.Slf4j;

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

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

@Api(tags = "课程相关接口")

@RestController

@RequestMapping("/lesson")

@Slf4j

public class LessonController {

@Autowired

private SessionUtils sessionUtils;

@Autowired

private LessonService lessonService;

/\*\*

\* 获取课程列表

\* @return LessonListResponse

\*/

@GetMapping(value = "/getLessonList", produces = "application/json")

@ApiOperation(value = "获取课程列表", response = LessonListResponse.class)

@ApiImplicitParams({

@ApiImplicitParam(name = "bookNumber", value = "书籍编号", paramType = "query", dataType = "Integer")

})

public Result getLessonList(@RequestParam("bookNumber") Integer bookNumber) {

try {

return Result.success(lessonService.getLessonList(bookNumber));

} catch (Exception e) {

if (e instanceof MyException) {

return Result.result(((MyException) e).getEnumExceptionType());

}

return Result.fail(e.getMessage());

}

}

/\*\*

\* 获取具体课程信息

\* @return LessonDetailResponse

\*/

@GetMapping(value = "/getLessonDetail", produces = "application/json")

@ApiOperation(value = "获取具体课程信息", response = LessonDetailResponse.class)

@ApiImplicitParams({

@ApiImplicitParam(name = "unitId", value = "unitId", paramType = "query", dataType = "String"),

@ApiImplicitParam(name = "bookNumber", value = "书籍编号", paramType = "query", dataType = "Integer")

})

public Result getLessonDetail(@RequestParam("unitId") String unitId,

@RequestParam("bookNumber") Integer bookNumber) {

try {

return Result.success(lessonService.getLessonDetail(unitId, bookNumber));

} catch (Exception e) {

if (e instanceof MyException) {

return Result.result(((MyException) e).getEnumExceptionType());

}

return Result.fail(e.getMessage());

}

}

}

package fun.redamancyxun.chinese.backend.controller.lesson.response;

import io.swagger.annotations.ApiModel;

import io.swagger.annotations.ApiModelProperty;

import lombok.AllArgsConstructor;

import lombok.Builder;

import lombok.Data;

import lombok.NoArgsConstructor;

import lombok.extern.slf4j.Slf4j;

@Data

@NoArgsConstructor

@AllArgsConstructor

@Builder

@Slf4j

@ApiModel("课程具体信息")

public class LessonDetailResponse {

@ApiModelProperty("课程归属单元")

private String unitId;

@ApiModelProperty("课程具体内容")

private String text;

}

package fun.redamancyxun.chinese.backend.controller.lesson.response;

import io.swagger.annotations.ApiModel;

import io.swagger.annotations.ApiModelProperty;

import lombok.AllArgsConstructor;

import lombok.Builder;

import lombok.Data;

import lombok.NoArgsConstructor;

import lombok.extern.slf4j.Slf4j;

import java.util.HashMap;

import java.util.Map;

@Data

@NoArgsConstructor

@AllArgsConstructor

@Builder

@Slf4j

@ApiModel("课程列表信息")

public class LessonListResponse {

@ApiModelProperty("单元总数")

private Integer totalUnits;

@ApiModelProperty("单元课程列表")

private Map<String, Integer> lessons;

}

package fun.redamancyxun.chinese.backend.controller.socre;

import fun.redamancyxun.chinese.backend.common.Result;

import fun.redamancyxun.chinese.backend.controller.socre.request.AudioScoreRequest;

import fun.redamancyxun.chinese.backend.controller.socre.response.AudioScoreResponse;

import fun.redamancyxun.chinese.backend.controller.user.response.UserInfoResponse;

import fun.redamancyxun.chinese.backend.exception.MyException;

import fun.redamancyxun.chinese.backend.service.ScoreActionService;

import fun.redamancyxun.chinese.backend.service.ScoreService;

import fun.redamancyxun.chinese.backend.util.SessionUtils;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiImplicitParam;

import io.swagger.annotations.ApiImplicitParams;

import io.swagger.annotations.ApiOperation;

import lombok.extern.slf4j.Slf4j;

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

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

import org.springframework.web.multipart.MultipartFile;

import javax.validation.constraints.NotNull;

@Api(tags = "音频评分相关接口")

@RestController

@RequestMapping("/score")

@Slf4j

public class ScoreController {

@Autowired

private ScoreService scoreService;

@Autowired

private ScoreActionService scoreActionService;

/\*\*

\* 音频评分

\* @param audioFile

\* @param courseId

\* @param courseContent

\* @return audioScoreResponse

\*/

@PostMapping(value = "/score")

@ApiOperation(value = "音频评分", response = AudioScoreResponse.class)

@ApiImplicitParams({

@ApiImplicitParam(name = "audio", value = "音频文件", required = true, dataType = "MultipartFile"),

@ApiImplicitParam(name = "bookId", value = "书籍id", required = true, dataType = "String"),

@ApiImplicitParam(name = "courseId", value = "文章id", required = true, dataType = "String"),

@ApiImplicitParam(name = "courseContent", value = "文章内容", required = true, dataType = "String")

})

public Result sore(@NotNull @RequestParam("audio") MultipartFile audioFile,

@NotNull @RequestParam("bookId") String bookId,

@NotNull @RequestParam("courseId") String courseId,

@NotNull @RequestParam("courseContent") String courseContent) {

try {

AudioScoreRequest audioScoreRequest = new AudioScoreRequest(audioFile, bookId, courseId, courseContent);

Result result = Result.success(scoreService.score(audioScoreRequest));

System.out.println(result);

return result;

} catch (Exception e) {

e.printStackTrace();

if (e instanceof MyException) {

return Result.result(((MyException) e).getEnumExceptionType(), ((MyException) e).getErrorMsg());

}

return Result.fail(e.getMessage());

}

}

}

package fun.redamancyxun.chinese.backend.controller.socre.request;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiModel;

import io.swagger.annotations.ApiModelProperty;

import lombok.AllArgsConstructor;

import lombok.Builder;

import lombok.Data;

import lombok.NoArgsConstructor;

import lombok.extern.slf4j.Slf4j;

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

import org.springframework.web.multipart.MultipartFile;

@Data

@NoArgsConstructor

@AllArgsConstructor

@Builder

@Slf4j

@ApiModel("音频上传请求")

public class AudioScoreRequest {

@ApiModelProperty("音频文件")

private MultipartFile audioFile;

@ApiModelProperty("书籍id")

private String bookId;

@ApiModelProperty("文章id")

private String courseId;

@ApiModelProperty("文章内容")

private String courseContent;

}

package fun.redamancyxun.chinese.backend.controller.socre.response;

import com.baomidou.mybatisplus.annotation.TableField;

import com.baomidou.mybatisplus.annotation.TableId;

import fun.redamancyxun.chinese.backend.service.ScoreService;

import io.swagger.annotations.ApiModel;

import io.swagger.annotations.ApiModelProperty;

import lombok.AllArgsConstructor;

import lombok.Builder;

import lombok.Data;

import lombok.NoArgsConstructor;

import lombok.extern.slf4j.Slf4j;

@Data

@NoArgsConstructor

@AllArgsConstructor

@Builder

@Slf4j

@ApiModel("模型评分结果")

public class AudioScoreResponse {

@ApiModelProperty("语速")

private Integer speed;

@ApiModelProperty("停顿")

private Integer pause;

@ApiModelProperty("声母")

private Integer initialConsonants;

@ApiModelProperty("韵母")

private Integer finalVowels;

@ApiModelProperty("声调")

private Integer tones;

@ApiModelProperty("完整度")

private Integer completeness;

@ApiModelProperty("评语&改进建议")

private String advice;

}

package fun.redamancyxun.chinese.backend.controller.user;

import fun.redamancyxun.chinese.backend.common.Result;

import fun.redamancyxun.chinese.backend.controller.user.request.UpdateUserInfoRequest;

import fun.redamancyxun.chinese.backend.controller.user.response.UserInfoResponse;

import fun.redamancyxun.chinese.backend.exception.MyException;

import fun.redamancyxun.chinese.backend.service.UserService;

import fun.redamancyxun.chinese.backend.util.SessionUtils;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiImplicitParam;

import io.swagger.annotations.ApiImplicitParams;

import io.swagger.annotations.ApiOperation;

import lombok.extern.slf4j.Slf4j;

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

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

import org.springframework.web.multipart.MultipartFile;

import javax.validation.constraints.NotNull;

@Api(tags = "用户操作相关接口")

@RestController

@RequestMapping("/user")

@Slf4j

public class UserController {

@Autowired

private SessionUtils sessionUtils;

@Autowired

private UserService userService;

/\*\*

\* 登录

\* @param telephone

\* @param password

\* @param role

\* @return UserInfoResponse

\*/

@PostMapping(value = "/login", produces = "application/json")

@ApiOperation(value = "登录", response = UserInfoResponse.class)

@ApiImplicitParams({

@ApiImplicitParam(name = "telephone", value = "手机号", required = true, paramType = "query", dataType = "String"),

@ApiImplicitParam(name = "password", value = "密码", required = true, paramType = "query", dataType = "String"),

@ApiImplicitParam(name = "role", value = "角色", required = true, paramType = "query", dataType = "Integer")

})

public Result login(@NotNull @RequestParam("telephone") String telephone,

@NotNull @RequestParam("password") String password,

@NotNull @RequestParam("role") Integer role) {

try {

return Result.success(userService.login(telephone, password, role));

} catch (Exception e) {

if (e instanceof MyException) {

return Result.result(((MyException) e).getEnumExceptionType());

}

return Result.fail(e.getMessage());

}

}

/\*\*

\* 检测登录状态

\* @return Integer

\*/

@GetMapping(value = "/checkLogin", produces = "application/json")

public Result checkLogin() throws MyException {

try {

return Result.success(userService.checkLogin());

} catch (Exception e) {

if (e instanceof MyException) {

return Result.result(((MyException) e).getEnumExceptionType());

}

return Result.fail(e.getMessage());

}

}

/\*\*

\* 登出

\* @return UserInfoResponse

\*/

@PostMapping(value = "/logout", produces = "application/json")

@ApiOperation(value = "登出", response = UserInfoResponse.class)

public Result logout() {

try {

return Result.success(userService.logout());

} catch (Exception e) {

if (e instanceof MyException) {

return Result.result(((MyException) e).getEnumExceptionType());

}

return Result.fail(e.getMessage());

}

}

/\*\*

\* 注册

\* @param telephone

\* @param password

\* @param role

\* @return UserInfoResponse

\*/

@PostMapping(value = "/signup", produces = "application/json")

@ApiOperation(value = "注册", response = UserInfoResponse.class)

@ApiImplicitParams({

@ApiImplicitParam(name = "telephone", value = "手机号", required = true, paramType = "query", dataType = "String"),

@ApiImplicitParam(name = "password", value = "密码", required = true, paramType = "query", dataType = "String"),

@ApiImplicitParam(name = "role", value = "角色", required = true, paramType = "query", dataType = "Integer")

})

public Result signup(@NotNull @RequestParam("telephone") String telephone,

@NotNull @RequestParam("password") String password,

@NotNull @RequestParam("role") Integer role) {

try {

return Result.success(userService.signup(telephone, password, role));

} catch (Exception e) {

if (e instanceof MyException) {

return Result.result(((MyException) e).getEnumExceptionType());

}

return Result.fail(e.getMessage());

}

}

/\*\*

\* 获取用户信息

\* @return UserInfoResponse

\*/

@GetMapping(value = "/getUserInfo", produces = "application/json")

@ApiOperation(value = "获取用户信息", response = UserInfoResponse.class)

public Result getUserInfo() {

try {

return Result.success(userService.getUserInfo());

} catch (Exception e) {

if (e instanceof MyException) {

return Result.result(((MyException) e).getEnumExceptionType());

}

e.printStackTrace();

return Result.fail(e.getMessage());

}

}

/\*\*

\* 更新用户信息

\* @param updateUserInfoRequest

\* @return UserInfoResponse

\*/

@PostMapping(value = "/updateUserInfo", produces = "application/json")

@ApiOperation(value = "更新用户信息", response = UserInfoResponse.class)

public Result updateUserInfo(@NotNull @RequestBody UpdateUserInfoRequest updateUserInfoRequest) {

try {

return Result.success(userService.updateUserInfo(updateUserInfoRequest));

} catch (Exception e) {

if (e instanceof MyException) {

return Result.result(((MyException) e).getEnumExceptionType());

}

return Result.fail(e.getMessage());

}

}

/\*\*

\* 更新用户的建议信息

\* @return UserInfoResponse

\*/

@PostMapping(value = "/updateSuggestion", produces = "application/json")

@ApiOperation(value = "更新用户的建议信息")

public Result updateSuggestion() {

try {

return Result.success(userService.updateOneAdvice());

} catch (Exception e) {

if (e instanceof MyException) {

return Result.result(((MyException) e).getEnumExceptionType());

}

return Result.fail(e.getMessage());

}

}

/\*\*

\* 修改用户密码

\* @param oldPassword

\* @param newPassword

\* @return UserInfoResponse

\*/

@PostMapping(value = "/changePassword", produces = "application/json")

@ApiOperation(value = "修改用户密码", response = UserInfoResponse.class)

@ApiImplicitParams({

@ApiImplicitParam(name = "oldPassword", value = "旧密码", required = true, paramType = "query", dataType = "String"),

@ApiImplicitParam(name = "newPassword", value = "新密码", required = true, paramType = "query", dataType = "String")

})

public Result changePassword(@NotNull @RequestParam("oldPassword") String oldPassword,

@NotNull @RequestParam("newPassword") String newPassword) {

try {

return Result.success(userService.changePassword(oldPassword, newPassword));

} catch (Exception e) {

if (e instanceof MyException) {

return Result.result(((MyException) e).getEnumExceptionType());

}

return Result.fail(e.getMessage());

}

}

/\*\*

\* 上传用户头像

\* @param file

\* @return String

\*/

@PostMapping(value = "/uploadPortrait", produces = "application/json")

@ApiOperation(value = "上传用户头像", response = String.class)

@ApiImplicitParam(name = "file", value = "图片文件", required = true, paramType = "formData", dataType = "file")

public Result uploadPortrait(@RequestParam("file") MultipartFile file) {

try {

return Result.success(userService.uploadPortrait(file));

} catch (Exception e) {

if (e instanceof MyException) {

return Result.result(((MyException) e).getEnumExceptionType());

}

return Result.fail(e.toString());

}

}

}

package fun.redamancyxun.chinese.backend.controller.user.request;

import com.baomidou.mybatisplus.annotation.TableField;

import io.swagger.annotations.ApiModel;

import io.swagger.annotations.ApiModelProperty;

import lombok.AllArgsConstructor;

import lombok.Builder;

import lombok.Data;

import lombok.NoArgsConstructor;

import lombok.extern.slf4j.Slf4j;

@Data

@NoArgsConstructor

@AllArgsConstructor

@Builder

@Slf4j

@ApiModel("更新用户信息请求")

public class UpdateUserInfoRequest {

@ApiModelProperty("用户名")

private String username;

@ApiModelProperty("性别(0男,1女)")

private Integer gender;

@ApiModelProperty("年龄")

@TableField(value = "age")

private Integer age;

@ApiModelProperty("国籍")

private String nation;

}

package fun.redamancyxun.chinese.backend.controller.user.response;

import com.baomidou.mybatisplus.annotation.TableField;

import com.baomidou.mybatisplus.annotation.TableId;

import fun.redamancyxun.chinese.backend.entity.ScoreAction;

import fun.redamancyxun.chinese.backend.entity.User;

import io.swagger.annotations.ApiModel;

import io.swagger.annotations.ApiModelProperty;

import lombok.AllArgsConstructor;

import lombok.Builder;

import lombok.Data;

import lombok.NoArgsConstructor;

import lombok.extern.slf4j.Slf4j;

import java.util.List;

@Data

@NoArgsConstructor

@AllArgsConstructor

@Builder

@Slf4j

@ApiModel("详细用户信息")

public class UserInfoResponse {

@ApiModelProperty("id")

private String id;

@ApiModelProperty("用户名")

private String username;

@ApiModelProperty("性别(0男,1女)")

private Integer gender;

@ApiModelProperty("年龄")

private Integer age;

@ApiModelProperty("国籍")

private String nation;

@ApiModelProperty("手机号")

private String telephone;

@ApiModelProperty("头像")

private String portrait;

@ApiModelProperty("身份(0是学生，1是老师)")

private Integer role;

@ApiModelProperty("发音特点、学习建议")

private String advice;

@ApiModelProperty("历史练习与评价记录")

private List<ScoreAction> history;

@ApiModelProperty("SessionId")

private String sessionId;

public UserInfoResponse(User user, List<ScoreAction> history) {

this.id = user.getId();

this.username = user.getUsername();

this.gender = user.getGender();

this.age = user.getAge();

this.nation = user.getNation();

this.telephone = user.getTelephone();

this.portrait = user.getPortrait();

this.role = user.getRole();

this.advice = user.getAdvice();

this.history = history;

this.sessionId = null;

}

public UserInfoResponse(User user) {

this.id = user.getId();

this.username = user.getUsername();

this.gender = user.getGender();

this.age = user.getAge();

this.nation = user.getNation();

this.telephone = user.getTelephone();

this.portrait = user.getPortrait();

this.role = user.getRole();

this.advice = user.getAdvice();

this.history = null;

this.sessionId = null;

}

public UserInfoResponse(User user, String sessionId) {

this.id = user.getId();

this.username = user.getUsername();

this.gender = user.getGender();

this.age = user.getAge();

this.nation = user.getNation();

this.telephone = user.getTelephone();

this.portrait = user.getPortrait();

this.role = user.getRole();

this.advice = user.getAdvice();

this.history = null;

this.sessionId = sessionId;

}

}

package fun.redamancyxun.chinese.backend.dto;

import fun.redamancyxun.chinese.backend.entity.ScoreAction;

import fun.redamancyxun.chinese.backend.entity.User;

import fun.redamancyxun.chinese.backend.exception.EnumExceptionType;

import fun.redamancyxun.chinese.backend.exception.MyException;

import io.swagger.annotations.ApiModel;

import io.swagger.annotations.ApiModelProperty;

import lombok.AllArgsConstructor;

import lombok.Builder;

import lombok.Data;

import lombok.NoArgsConstructor;

import java.io.Serializable;

import java.util.List;

/\*\*

\* session缓存实体

\*/

@Data

@NoArgsConstructor

@AllArgsConstructor

@Builder

@ApiModel("SessionData 会话实体")

public class SessionData implements Serializable {

@ApiModelProperty("id")

private String id;

@ApiModelProperty("用户名")

private String username;

@ApiModelProperty("性别(0男,1女)")

private Integer gender;

@ApiModelProperty("年龄")

private Integer age;

@ApiModelProperty("国籍")

private String nation;

@ApiModelProperty("手机号")

private String telephone;

@ApiModelProperty("头像")

private String portrait;

@ApiModelProperty("身份(0是学生，1是老师)")

private Integer role;

@ApiModelProperty("发音特点、学习建议")

private String advice;

@ApiModelProperty("历史练习与评价记录")

private List<ScoreAction> history;

public SessionData(User user, List<ScoreAction> history) {

this.id = user.getId();

this.username = user.getUsername();

this.gender = user.getGender();

this.age = user.getAge();

this.nation = user.getNation();

this.telephone = user.getTelephone();

this.portrait = user.getPortrait();

this.role = user.getRole();

this.advice = user.getAdvice();

this.history = history;

}

public SessionData(User user) {

this.id = user.getId();

this.username = user.getUsername();

this.gender = user.getGender();

this.age = user.getAge();

this.nation = user.getNation();

this.telephone = user.getTelephone();

this.portrait = user.getPortrait();

this.role = user.getRole();

this.advice = user.getAdvice();

this.history = null;

}

}

package fun.redamancyxun.chinese.backend.entity;

import com.baomidou.mybatisplus.annotation.\*;

import io.swagger.annotations.ApiModel;

import io.swagger.annotations.ApiModelProperty;

import lombok.AllArgsConstructor;

import lombok.Builder;

import lombok.Data;

import lombok.NoArgsConstructor;

import javax.persistence.Id;

import java.time.LocalDateTime;

@ApiModel("advice\_action “发育特点、学习建议”\_记录")

@TableName(value ="advice\_action")

@Data

@NoArgsConstructor

@AllArgsConstructor

@Builder

public class AdviceAction {

@Id

@ApiModelProperty("id")

@TableId(value = "id", type = IdType.AUTO)

private Integer id;

@ApiModelProperty("受评用户id")

@TableField(value = "user\_id")

private String userId;

@ApiModelProperty("发育特点、学习建议")

@TableField(value = "advice")

private String advice;

@ApiModelProperty("创建时间")

@TableField(value = "create\_time",fill = FieldFill.INSERT)

private LocalDateTime createTime;

}

package fun.redamancyxun.chinese.backend.entity;

import com.baomidou.mybatisplus.annotation.IdType;

import com.baomidou.mybatisplus.annotation.TableField;

import com.baomidou.mybatisplus.annotation.TableId;

import com.baomidou.mybatisplus.annotation.TableName;

import io.swagger.annotations.ApiModel;

import io.swagger.annotations.ApiModelProperty;

import lombok.AllArgsConstructor;

import lombok.Builder;

import lombok.Data;

import lombok.NoArgsConstructor;

import javax.persistence.Id;

import java.time.OffsetDateTime;

@ApiModel("score\_action 评分记录")

@TableName(value ="score\_action")

@Data

@NoArgsConstructor

@AllArgsConstructor

@Builder

public class ScoreAction {

@Id

@ApiModelProperty("id")

@TableId(value = "id", type = IdType.AUTO)

private Integer id;

@ApiModelProperty("受评用户id")

@TableField(value = "user\_id")

private String userId;

@ApiModelProperty("受评书籍id")

@TableField(value = "book\_id")

private String bookId;

double completenessScore = headerRow.getCell(5).getNumericCellValue();

double phoneticScore = headerRow.getCell(6).getNumericCellValue();

double toneScore = headerRow.getCell(7).getNumericCellValue();

double totalScore = headerRow.getCell(8).getNumericCellValue();

// 生成txt文件，文件名与xlsx文件同名

String outputFileName = fileName.substring(0, fileName.lastIndexOf('.')) + ".txt";

File outputFile = new File(savePath + outputFileName);

try (BufferedWriter writer = new BufferedWriter(new FileWriter(outputFile))) {

// 在txt最前面加上一段话

String introText = "此数据为留学生音频量化数据，每条数据都是口语音频中的音素信息，每条数据格式为：{content:{beg\_pos,end\_pos,dp\_message,mono\_tone,is\_yun,perr\_msg};}（数据为空代表数据无此属性）。参数说明：{'content':'音素内容','beg\_pos':'开始边界时间','end\_pos':'结束边界时间','dp\_message':'0正常；16漏读；32增读；64回读；128替换（当dp\_message不为0时，perr\_msg可能出现与dp\_message值保持一致的情况）','mono\_tone':'调型','is\_yun':'0声母，1韵母','perr\_msg':'当is\_yun=0时，perr\_msg有两种状态：0声母正确，1声母错误；当is\_yun=1时，perr\_msg有四种状态：0韵母和调型均正确，1韵母错误，2调型错误，3韵母和调型均错误'}（注：content为sil表明是非文本中有的内容）。其中给出的第一条数据是可参考的讯飞AI维度评分。量化数据如下：{"

+ "{" + "流畅度分:" + smoothnessScore + ",完整度分:" + completenessScore + ",声韵分:" + phoneticScore + ",调型分:" + toneScore + ",总分【模型回归】:" + totalScore + "};";

writer.write(introText);

// 从第二行开始读取数据

for (int i = 1; i <= sheet.getLastRowNum(); i++) {

Row row = sheet.getRow(i);

String content = row.getCell(1).getStringCellValue();

List<String> cToKValues = new ArrayList<>();

for (int j : new int[]{2, 3, 9, 10, 11, 12}) {

Cell cell = row.getCell(j);

if (cell != null) {

if (cell.getCellType() == CellType.NUMERIC) {

cToKValues.add(String.valueOf((int) cell.getNumericCellValue()));

} else {

cToKValues.add(cell.getStringCellValue());

}

} else {

cToKValues.add("");

}

}

String contentLine = content + ":{" + String.join(",", cToKValues) + "};";

writer.write(contentLine);

}

writer.write("}");

}

} catch (IOException e) {

e.printStackTrace();

}

System.out.println("处理完成，结果已保存到 " + fileName.substring(0, fileName.lastIndexOf('.')) + ".txt" + " 文件中。");

}

}

}

package fun.redamancyxun.chinese.backend.score;

import org.apache.commons.io.FilenameUtils;

import org.apache.poi.ss.usermodel.\*;

import org.apache.poi.xssf.usermodel.XSSFWorkbook;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import org.w3c.dom.NodeList;

import org.xml.sax.SAXException;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.ParserConfigurationException;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

public class GenerateQuantitativeXlsx {

public static void main(String[] args) {

try {

// Load Excel file

FileInputStream excelFile = new FileInputStream(new File("src/main/java/fun/redamancyxun/chinese/backend/score/AudioTextMapping.xlsx"));

Workbook workbook = new XSSFWorkbook(excelFile);

Sheet sheet = workbook.getSheetAt(0);

// Map PCM paths to text

Map<String, String> audioTextMapping = new HashMap<>();

for (Row row : sheet) {

if (row.getRowNum() == 0) continue; // Skip header

String pcmPath = row.getCell(0).getStringCellValue();

String text = row.getCell(1).getStringCellValue();

audioTextMapping.put(pcmPath, text);

}

// Process each XML file

for (String pcmPath : audioTextMapping.keySet()) {

String xmlFilename = (FilenameUtils.removeExtension(pcmPath) + ".xml").replace("pcm", "xml");

System.out.println("正在处理 " + xmlFilename);

// Parse XML file

File xmlFile = new File(xmlFilename);

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.parse(xmlFile);

// Normalize the document

doc.getDocumentElement().normalize();

// Collect data from XML

List<Map<String, String>> allData = new ArrayList<>();

NodeList readSentenceNodes = doc.getElementsByTagName("read\_sentence");

for (int temp = 0; temp < readSentenceNodes.getLength(); temp++) {

Element readSentence = (Element) readSentenceNodes.item(temp);

if (readSentence.getAttribute("content").isEmpty()) continue;

Map<String, String> readSentenceData = new HashMap<>();

readSentenceData.put("filename", xmlFilename);

readSentenceData.put("content", readSentence.getAttribute("content"));

readSentenceData.put("beg\_pos", readSentence.getAttribute("beg\_pos"));

readSentenceData.put("end\_pos", readSentence.getAttribute("end\_pos"));

readSentenceData.put("fluency\_score", readSentence.getAttribute("fluency\_score"));

readSentenceData.put("integrity\_score", readSentence.getAttribute("integrity\_score"));

readSentenceData.put("phone\_score", readSentence.getAttribute("phone\_score"));

readSentenceData.put("tone\_score", readSentence.getAttribute("tone\_score"));

readSentenceData.put("total\_score", readSentence.getAttribute("total\_score"));

allData.add(readSentenceData);

NodeList sentenceNodes = readSentence.getElementsByTagName("sentence");

for (int i = 0; i < sentenceNodes.getLength(); i++) {

Element sentence = (Element) sentenceNodes.item(i);

NodeList wordNodes = sentence.getElementsByTagName("word");

for (int j = 0; j < wordNodes.getLength(); j++) {

Element word = (Element) wordNodes.item(j);

NodeList syllNodes = word.getElementsByTagName("syll");

for (int k = 0; k < syllNodes.getLength(); k++) {

Element syll = (Element) syllNodes.item(k);

NodeList phoneNodes = syll.getElementsByTagName("phone");

for (int l = 0; l < phoneNodes.getLength(); l++) {

Element phone = (Element) phoneNodes.item(l);

Map<String, String> phoneData = new HashMap<>();

phoneData.put("filename", xmlFilename);

phoneData.put("content", phone.getAttribute("content"));

phoneData.put("beg\_pos", phone.getAttribute("beg\_pos"));

phoneData.put("end\_pos", phone.getAttribute("end\_pos"));

phoneData.put("dp\_message", phone.getAttribute("dp\_message"));

phoneData.put("mono\_tone", phone.getAttribute("mono\_tone"));

phoneData.put("is\_yun", phone.getAttribute("is\_yun"));

phoneData.put("perr\_msg", phone.getAttribute("perr\_msg"));

allData.add(phoneData);

}

}

}

}

}

// Save data to Excel

Workbook excelWorkbook = new XSSFWorkbook();

Sheet excelSheet = excelWorkbook.createSheet("Data");

int rowCount = 0;

for (Map<String, String> data : allData) {

Row row = excelSheet.createRow(rowCount++);

int colCount = 0;

for (String key : data.keySet()) {

Cell cell = row.createCell(colCount++);

cell.setCellValue(data.get(key));

}

}

try (FileOutputStream outputStream = new FileOutputStream((FilenameUtils.removeExtension(xmlFilename) + ".xlsx").replace("xml", "xlsx"))) {

excelWorkbook.write(outputStream);

}

}

} catch (IOException | ParserConfigurationException | SAXException e) {

e.printStackTrace();

}

}

}

package fun.redamancyxun.chinese.backend.score;

import com.google.gson.Gson;

import com.google.gson.JsonObject;

import lombok.AllArgsConstructor;

import lombok.NoArgsConstructor;

import okhttp3.HttpUrl;

import okhttp3.OkHttpClient;

import okhttp3.Request;

import okhttp3.Response;

import okhttp3.WebSocket;

import okhttp3.WebSocketListener;

import org.apache.poi.ss.formula.functions.Count;

import org.apache.poi.ss.usermodel.Row;

import org.apache.poi.ss.usermodel.Sheet;

import org.apache.poi.ss.usermodel.Workbook;

import org.apache.poi.ss.usermodel.WorkbookFactory;

import org.hamcrest.core.Is;

import org.springframework.stereotype.Service;

import javax.crypto.Mac;

import javax.crypto.spec.SecretKeySpec;

import java.io.\*;

import java.net.URL;

import java.nio.channels.FileChannel;

import java.nio.charset.Charset;

import java.nio.charset.StandardCharsets;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.nio.file.StandardOpenOption;

import java.text.SimpleDateFormat;

import java.util.\*;

import java.util.concurrent.CountDownLatch;

/\*\*

\* 语言测评工具类

\*/

@AllArgsConstructor

public class IseDemo extends WebSocketListener {

/\*\*

\* 测评平台常量

\*/

private static final String hostUrl ="https://ise-api.xfyun.cn/v2/open-ise"; // 开放评测地址

private static final String appid = "297c85b1"; // 控制台获取

private static final String apiSecret = "MzFiZDM0YWY3OWRmNjFmZWY2ZDhmYjY3"; // 控制台获取

private static final String apiKey = "0bd876a7866a314f1dbe9a1024eccd2f"; // 控制台获取

/\*\*

\* 测评过程参数

\*/

private static final String sub= "ise"; // 服务类型sub,开放评测值为ise

private static final String ent= "cn\_vip"; // 语言标记参数 ent(cn\_vip中文,en\_vip英文)

/\*\*

\* 音频流状态

\*/

private static final int StatusFirstFrame = 0; // 第一帧

private static final int StatusContinueFrame = 1; // 中间帧

private static final int StatusLastFrame = 2; // 最后一帧

private static final Base64.Encoder encoder = Base64.getEncoder(); // 编码

private static final Base64.Decoder decoder = Base64.getDecoder(); // 解码

private static final Gson json = new Gson();

/\*private int aus = 1;

private static Date dateBegin = new Date();// 开始时间

private static Date dateEnd = new Date();// 结束时间

private static final SimpleDateFormat sdf = new SimpleDateFormat("yyy-MM-dd HH:mm:ss.SSS");\*/

private static long beginTime = (new Date()).getTime();

private static long endTime = (new Date()).getTime();

/\*\*

\* 需要个性化操作的成员变量

\*/

// 题型、文本、音频要请注意做同步变更(如果是英文评测,请注意变更ent参数的值)

private static String category; // 题型

private static String text; // 评测试题,英文试题:[content]\nthere was a gentleman live near my house.

private static String file; // 评测音频,如传mp3格式请改变参数aue的值为lame

/\*\*

\* 输出xml结果

\*/

public static String xmlData;

/\*\*

\* 线程等待

\*/

public static CountDownLatch latch;

public static void iseDemoScore(String text, String file) throws Exception {

System.out.println("即将评测文件是：" + file + "\n文本是：" + text);

IseDemo.category = "read\_sentence";

IseDemo.text = text;

IseDemo.file = file;

String authUrl = getAuthUrl(hostUrl, apiKey, apiSecret); // 构建鉴权url

OkHttpClient client = new OkHttpClient.Builder().build();

System.out.println(authUrl);

String url = authUrl.replace("http://", "ws://").replace("https://", "wss://"); // 将url中的 schema http://和https://分别替换为ws:// 和 wss://

Request request = new Request.Builder().url(url).build();

System.out.println("url===>" + url);

WebSocket webSocket = client.newWebSocket(request, new IseDemo());

}

public static void iseDemoScore(String category, String text, String file) throws Exception {

System.out.println("即将评测文件是：" + file + "\n文本是：" + text);

IseDemo.category = category;

IseDemo.text = text;

IseDemo.file = file;

String authUrl = getAuthUrl(hostUrl, apiKey, apiSecret); // 构建鉴权url

OkHttpClient client = new OkHttpClient.Builder().build();

System.out.println(authUrl);

String url = authUrl.replace("http://", "ws://").replace("https://", "wss://"); // 将url中的 schema http://和https://分别替换为ws:// 和 wss://

Request request = new Request.Builder().url(url).build();

System.out.println("url===>" + url);

WebSocket webSocket = client.newWebSocket(request, new IseDemo());

}

// public static void main(String[] args) throws Exception {

// Map<String, String> audioTextMapping = readAudioTextMapping("src/main/java/fun/redamancyxun/chinese/backend/score/AudioTextMapping.xlsx");

// for (Map.Entry<String, String> entry : audioTextMapping.entrySet()) {

// try {

// IseDemo.iseDemoScore(entry.getValue(), entry.getKey());

// } catch (Exception e) {

// e.printStackTrace();

// throw e;

// }

// }

// }

// 读取Excel文件获取音频-文本映射

private static Map<String, String> readAudioTextMapping(String excelFilePath) throws IOException {

Map<String, String> mapping = new HashMap<>();

FileInputStream fileInputStream = new FileInputStream(excelFilePath);

Workbook workbook = WorkbookFactory.create(fileInputStream);

Sheet sheet = workbook.getSheetAt(0);

for (Row row : sheet) {

if (row.getRowNum() == 0) continue; // 跳过标题行

String pcmPath = row.getCell(0).getStringCellValue();

String text = row.getCell(1).getStringCellValue();

mapping.put(pcmPath, text);

}

workbook.close();

fileInputStream.close();

return mapping;

}

// WebSocket握手连接并上传音频数据

@Override

public void onOpen(WebSocket webSocket, Response response) {

super.onOpen(webSocket, response);

new Thread(() -> {

// 连接成功，开始发送数据

int frameSize = 1280; // 每一帧音频的大小,建议每 40ms 发送 1280B，大小可调整，但是不要超过19200B，即base64压缩后能超过26000B，否则会报错10163数据过长错误

int intervel = 40;

int status = 0; // 音频的状态

// FileInputStream fs = new FileInputStream("0.pcm");

ssb(webSocket);

// ttp(webSocket);

beginTime=(new Date()).getTime();

try (FileInputStream fs = new FileInputStream(file)) {

byte[] buffer = new byte[frameSize];

// 发送音频

end:

while (true) {

int len = fs.read(buffer);

if (len == -1) {

status = StatusLastFrame; // 文件读完，改变status 为 2

}

switch (status) {

case StatusFirstFrame: // 第一帧音频status = 0

send(webSocket,1,1,Base64.getEncoder().encodeToString(Arrays.copyOf(buffer, len)));

status=StatusContinueFrame;// 中间帧数

break;

case StatusContinueFrame: // 中间帧status = 1

send(webSocket,2,1,Base64.getEncoder().encodeToString(Arrays.copyOf(buffer, len)));

break;

case StatusLastFrame: // 最后一帧音频status = 2 ，标志音频发送结束

send(webSocket,4,2,"");

System.out.println("sendlast");

endTime=(new Date()).getTime();

System.out.println("总耗时："+(endTime-beginTime)+"ms");

break end;

}

Thread.sleep(intervel); // 模拟音频采样延时

}

System.out.println("all data is send");

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

} catch (Exception e) {

e.printStackTrace();

}

}).start();

}

// 上传参数添加与发送

private void ssb(WebSocket webSocket) {

ParamBuilder p = new ParamBuilder();

p.add("common", new ParamBuilder()

.add("app\_id", appid)

)

.add("business", new ParamBuilder()

.add("category", category)

.add("rstcd", "utf8")

//群体：adult成人 、youth（中学，效果与设置pupil参数一致）、pupil小学

//仅中文字、词、句题型支持

.add("group", "adult")

//打分门限值：hard、common、easy

//仅英文引擎支持

//.add("check\_type","common")

//学段：junior(1,2年级) middle(3,4年级) senior(5,6年级)

//仅中文题型：中小学的句子、篇章题型支持

//.add("grade","junior")

//extra\_ability生效条件：ise\_unite=1,rst=entirety

//.add("ise\_unite","1")

//.add("rst","entirety")

/\*1.全维度(准确度分、流畅度分、完整度打分) ,extra\_ability值为multi\_dimension

2.支持因素错误信息显示(声韵、调型是否正确),extra\_ability值为syll\_phone\_err\_msg

3.单词基频信息显示（基频开始值、结束值）,extra\_ability值为pitch ，仅适用于单词和句子题型

4.(字词句篇均适用,如选多个能力，用分号;隔开如:syll\_phone\_err\_msg;pitch;multi\_dimension)\*/

//.add("extra\_ability","multi\_dimension")

//试卷部分添加拼音,限制条件：添加拼音的汉字个数不超过整个试卷中汉字个数的三分之一。

//jin1|tian1|天气怎么样支持

//分制转换，rst=entirety是默认值，请根据文档推荐选择使用百分制

.add("sub",sub)

.add("ent",ent)

.add("tte", "utf-8")

.add("cmd", "ssb")

.add("auf", "audio/L16;rate=16000")

.add("aue", "raw")

//评测文本(new String(new byte[] { (byte) 0xEF, (byte) 0xBB, (byte) 0xBF })+text)

.add("text",'\uFEFF'+text)//Base64.getEncoder().encodeToString(text.getBytes())

).add("data", new ParamBuilder()

.add("status", 0)

.add("data", ""));

//System.err.println(p.toString());

webSocket.send(p.toString());

}

// 客户端给服务端发送数据

public void send(WebSocket webSocket, int aus,int status, String data) {

ParamBuilder p = new ParamBuilder();

p.add("business", new ParamBuilder()

.add("cmd", "auw")

.add("aus", aus)

.add("aue", "raw")

).add("data",new ParamBuilder()

.add("status",status)

.add("data",data)

.add("data\_type",1)

.add("encoding","raw")

);

//System.out.println("发送的数据"+p.toString());

webSocket.send(p.toString());

}

// 客户端接收服务端消息

@Override

public void onMessage(WebSocket webSocket, String text) {

super.onMessage(webSocket, text);

//System.out.println(text);

IseNewResponseData resp = json.fromJson(text, IseNewResponseData.class);

if (resp != null) {

if (resp.getCode() != 0) {

System.out.println("code=>" + resp.getCode() + " error=>" + resp.getMessage() + " sid=" + resp.getSid());

System.out.println("错误码查询链接：https://www.xfyun.cn/document/error-code");

return;

}

if (resp.getData() != null) {

if (resp.getData().getData() != null) {

//中间结果处理

}

if (resp.getData().getStatus() == 2) {

try {

System.out.println("sid:" + resp.getSid() + " 最终识别结果" + new String(decoder.decode(resp.getData().getData()), "UTF-8"));

// 解码Base64数据并转换为字符串

IseDemo.xmlData = new String(decoder.decode(resp.getData().getData()), "UTF-8");

// // 将XML数据保存到文件中

// String pcmFilename = file; // 假设这是音频文件名

// String xmlFilename = pcmFilename.replace(".pcm", ".xml");

// String xmlFilePath = xmlFilename.replace("pcm", "xml");

//

// // 创建文件并确保目录存在

// Path path = Paths.get(xmlFilePath);

// Files.createDirectories(path.getParent()); // 确保目录存在

// Files.createFile(path); // 创建文件

//

// // 使用try-with-resources确保文件流正确关闭

// try (BufferedWriter writer = new BufferedWriter(new FileWriter(xmlFilePath))) {

// writer.write(xmlData);

// }

//

// // 强制将数据写入磁盘

// FileChannel fileChannel = FileChannel.open(path, StandardOpenOption.WRITE);

// fileChannel.force(true); // true 表示同时刷新元数据

// fileChannel.close();

//

// System.out.println("XML result saved as " + xmlFilePath);

// webSocket.close(1000, "");

} catch (Exception e) {

e.printStackTrace();

}

webSocket.close(1000, "");

// 减少 CountDownLatch 的计数

latch.countDown();

} else {

// todo 根据返回的数据处理

}

}

}

}

// 鉴权

public static String getAuthUrl(String hostUrl, String apiKey, String apiSecret) throws Exception {

URL url = new URL(hostUrl);

SimpleDateFormat format = new SimpleDateFormat("EEE, dd MMM yyyy HH:mm:ss z", Locale.US);

format.setTimeZone(TimeZone.getTimeZone("GMT"));

String date = format.format(new Date());

//String date = format.format(new Date());

//System.err.println(date);

StringBuilder builder = new StringBuilder("host: ").append(url.getHost()).append("\n").//

append("date: ").append(date).append("\n").//

append("GET ").append(url.getPath()).append(" HTTP/1.1");

//System.err.println(builder);

Charset charset = Charset.forName("UTF-8");

Mac mac = Mac.getInstance("hmacsha256");

SecretKeySpec spec = new SecretKeySpec(apiSecret.getBytes(charset), "hmacsha256");

mac.init(spec);

byte[] hexDigits = mac.doFinal(builder.toString().getBytes(charset));

String sha = Base64.getEncoder().encodeToString(hexDigits);

//System.err.println(sha);

String authorization = String.format("api\_key=\"%s\", algorithm=\"%s\", headers=\"%s\", signature=\"%s\"", apiKey, "hmac-sha256", "host date request-line", sha);

//System.err.println(authorization);

HttpUrl httpUrl = HttpUrl.parse("https://" + url.getHost() + url.getPath()).newBuilder().//

addQueryParameter("authorization", Base64.getEncoder().encodeToString(authorization.getBytes(charset))).//

addQueryParameter("date", date).//

addQueryParameter("host", url.getHost()).//

build();

return httpUrl.toString();

}

// JSON解析

private static class IseNewResponseData{

private int code;

private String message;

private String sid;

private Data data;

public int getCode() {

return code;

}

public void setCode(int code) {

this.code = code;

}

public String getMessage() {

return message;

}

public void setMessage(String message) {

this.message = message;

}

public String getSid() {

return sid;

}

public void setSid(String sid) {

this.sid = sid;

}

public Data getData() {

return data;

}

public void setData(Data data) {

this.data = data;

}

}

private static class Data {

private int status;

private String data;

public int getStatus() {

return status;

}

public void setStatus(int status) {

this.status = status;

}

public String getData() {

return data;

}

public void setData(String data) {

this.data = data;

}

}

// 传参构建

public static class ParamBuilder {

private JsonObject jsonObject = new JsonObject();

public ParamBuilder add(String key, String val) {

this.jsonObject.addProperty(key, val);

return this;

}

public ParamBuilder add(String key, int val) {

this.jsonObject.addProperty(key, val);

return this;

}

public ParamBuilder add(String key, boolean val) {

this.jsonObject.addProperty(key, val);

return this;

}

public ParamBuilder add(String key, float val) {

this.jsonObject.addProperty(key, val);

return this;

}

public ParamBuilder add(String key, JsonObject val) {

this.jsonObject.add(key, val);

return this;

}

public ParamBuilder add(String key, ParamBuilder val) {

this.jsonObject.add(key, val.jsonObject);

return this;

}

@Override

public String toString() {

return this.jsonObject.toString();

}

}

}

package fun.redamancyxun.chinese.backend.service;

import fun.redamancyxun.chinese.backend.entity.AdviceAction;

import java.util.List;

/\*\*

\* 建议操作相关接口

\* @author Redamancy

\* @description 建议行为相关接口

\* @createDate 2024-11-9 22:39:04

\*/

public interface AdviceActionService {

// 创建一条建议记录

AdviceAction createAdviceAction(String userId, String advice);

// 根据用户id获取该用户的建议记录

List<AdviceAction> findAdviceActionByUserId(String userId);

}

package fun.redamancyxun.chinese.backend.service;

//import fun.redamancyxun.chinese.backend.controller.forum.request.ContactRequest;

import org.springframework.web.multipart.MultipartFile;

/\*\*

\* @author Redamancy

\* @description 全局服务

\* @createDate 2024-04-03 22:39:04

\*/

public interface GlobalService {

// 上传图片

String uploadImage(MultipartFile file);

// // 联系我们

// Boolean contact(ContactRequest contactRequest);

}

package fun.redamancyxun.chinese.backend.service;

import fun.redamancyxun.chinese.backend.controller.lesson.response.LessonDetailResponse;

import fun.redamancyxun.chinese.backend.controller.lesson.response.LessonListResponse;

/\*\*

\* 课程相关接口

\* @author Redamancy

\* @description 课程相关接口

\* @createDate 2024-10-16 22:39:04

\*/

public interface LessonService {

// 获取课程详情

LessonDetailResponse getLessonDetail(String unitId, Integer bookNumber);

// 获取课程列表

LessonListResponse getLessonList(Integer bookNumber);

}

package fun.redamancyxun.chinese.backend.service;

import com.baomidou.mybatisplus.annotation.TableField;

import fun.redamancyxun.chinese.backend.entity.ScoreAction;

import io.swagger.annotations.ApiModelProperty;

import java.util.List;

/\*\*

\* 评分操作相关接口

\* @author Redamancy

\* @description 评分行为相关接口

\* @createDate 2024-10-16 22:39:04

\*/

public interface ScoreActionService {

// 创建评分行为

ScoreAction createScoreAction(String userId,String bookId, String textId, Integer speed, Integer pause, Integer initialConsonants, Integer finalVowels, Integer tones, Integer completeness, String advice);

// 查找用户的评分记录

List<ScoreAction> findScoreActionByUserId(String userId);

}

package fun.redamancyxun.chinese.backend.service;

import fun.redamancyxun.chinese.backend.controller.socre.request.AudioScoreRequest;

import fun.redamancyxun.chinese.backend.controller.socre.response.AudioScoreResponse;

import org.springframework.stereotype.Service;

import java.io.IOException;

/\*\*

\* 评分相关接口

\* @author Redamancy

\* @description 评分相关接口

\* @createDate 2024-11-3 22:39:04

\*/

public interface ScoreService {

// 将字节流byte[]保存为pcm文件

String byte2pcm(byte[] data);

// 利用pcm文件路径和文本内容生成音频-文本对照表

void pcm2xlsx(String pcmPath, String text);

// 利用讯飞模型，量化pcm音频数据为xml数据

void pcm2xml() throws Exception;

// 利用python脚本，将xml文件转化为txt

void xml2txt();

// 利用训练好的模型，整合信息并得出音频得分

AudioScoreResponse score(AudioScoreRequest audioScoreRequest) throws Exception;

}

package fun.redamancyxun.chinese.backend.service;

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

import fun.redamancyxun.chinese.backend.controller.user.request.UpdateUserInfoRequest;

import fun.redamancyxun.chinese.backend.controller.user.response.UserInfoResponse;

import fun.redamancyxun.chinese.backend.entity.User;

import fun.redamancyxun.chinese.backend.exception.MyException;

import org.springframework.web.multipart.MultipartFile;

/\*\*

\* 针对表【user(用户信息)】的数据库操作Service

\*/

public interface UserService extends IService<User> {

// 登录

UserInfoResponse login(String telephone, String password, Integer role) throws Exception;

// 检测登录状态

Integer checkLogin() throws MyException;

// 注销

UserInfoResponse logout();

// 注册

UserInfoResponse signup(String telephone, String password, Integer role) throws MyException;

// 获取用户信息

UserInfoResponse getUserInfo();

// 更新用户信息

UserInfoResponse updateUserInfo(UpdateUserInfoRequest updateUserInfoRequest);

// 更新一个用户的“发育特点、学习建议”

UserInfoResponse updateOneAdvice(String userId);

// 更新当前用户的“发育特点、学习建议”

UserInfoResponse updateOneAdvice();

// 更新所有用户的“发育特点、学习建议”

UserInfoResponse updateAllAdvice();

// 修改密码

UserInfoResponse changePassword(String oldPassword, String newPassword);

// 上传用户头像

String uploadPortrait(MultipartFile file);

// void deleteUser();

User getUserById(String userId);

}

package fun.redamancyxun.chinese.backend.service.impl;

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

import fun.redamancyxun.chinese.backend.entity.AdviceAction;

import fun.redamancyxun.chinese.backend.entity.User;

import fun.redamancyxun.chinese.backend.exception.EnumExceptionType;

import fun.redamancyxun.chinese.backend.exception.MyException;

import fun.redamancyxun.chinese.backend.mapper.AdviceActionMapper;

import fun.redamancyxun.chinese.backend.service.AdviceActionService;

import fun.redamancyxun.chinese.backend.service.UserService;

import fun.redamancyxun.chinese.backend.util.SessionUtils;

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

import org.springframework.stereotype.Service;

import java.time.LocalDateTime;

import java.util.List;

/\*\*

\* 建议操作相关接口

\* @author Redamancy

\* @description 建议行为相关接口

\* @createDate 2024-11-9 22:39:04

\*/

@Service

public class AdviceActionImpl implements AdviceActionService {

@Autowired

private UserService userService;

@Autowired

private AdviceActionMapper adviceActionMapper;

@Autowired

private SessionUtils sessionUtils;

/\*\*

\* 创建建议行为

\* @param userId 用户id

\* @param advice 建议内容

\* @return AdviceAction

\* @throws MyException 自定义异常

\*/

@Override

public AdviceAction createAdviceAction(String userId, String advice) throws MyException {

// 获取用户信息

User user = userService.getUserById(userId);

// sessionUtils.refreshData(user);

AdviceAction adviceAction = AdviceAction.builder()

.userId(user.getId())

.advice(advice)

.createTime(LocalDateTime.now())

.build();

if (adviceActionMapper.insert(adviceAction) == 0) {

throw new MyException(EnumExceptionType.INSERT\_FAILED);

}

return adviceAction;

}

/\*\*

\* 根据用户id查找建议行为

\* @param userId 用户id

\* @return AdviceAction

\* @throws MyException 自定义异常

\*/

@Override

public List<AdviceAction> findAdviceActionByUserId(String userId) throws MyException {

// 获取用户信息

User user = userService.getUserById(userId);

// sessionUtils.refreshData(user);

QueryWrapper<AdviceAction> queryWrapper = new QueryWrapper<>();

queryWrapper.eq("user\_id", user.getId());

return adviceActionMapper.selectList(queryWrapper);

}

}

//package fun.redamancyxun.chinese.backend.service.impl;

//

//import cn.hutool.core.io.FileUtil;

//import cn.hutool.core.util.IdUtil;

////import fun.redamancyxun.chinese.backend.controller.forum.request.ContactRequest;

//import fun.redamancyxun.chinese.backend.entity.User;

//import fun.redamancyxun.chinese.backend.exception.EnumExceptionType;

//import fun.redamancyxun.chinese.backend.exception.MyException;

//import fun.redamancyxun.chinese.backend.service.GlobalService;

//import fun.redamancyxun.chinese.backend.service.UserService;

//import fun.redamancyxun.chinese.backend.util.ImageUtil;

//import fun.redamancyxun.chinese.backend.util.MessageUtil;

//import fun.redamancyxun.chinese.backend.util.SessionUtils;

//import lombok.extern.slf4j.Slf4j;

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

//import org.springframework.beans.factory.annotation.Value;

//import org.springframework.mail.javamail.JavaMailSender;

//import org.springframework.stereotype.Service;

//import org.springframework.web.multipart.MultipartFile;

//

//import java.io.File;

//import java.io.IOException;

//

//import static fun.redamancyxun.chinese.backend.common.CommonConstants.USER\_FILE\_PATH;

//

///\*\*

// \* @author Redamancy

// \* @description 全局服务

// \* @createDate 2024-04-03 22:39:04

// \*/

//@Service

//@Slf4j

//public class GlobalServiceImpl implements GlobalService {

//

// @Autowired

// private SessionUtils sessionUtils;

//

// @Autowired

// private MessageUtil messageUtil;

//

// @Autowired

// private UserService userService;

//

// @Value("${spring.mail.username}")

// private String sender;

//

// /\*\*

// \* 上传图片

// \* @param file 图片文件

// \* @return 图片路径

// \* @throws MyException 通用异常

// \*/

// @Override

// public String uploadImage(MultipartFile file) throws MyException {

// //获取文件的原始文件名

// String original = file.getOriginalFilename();

// //生成一个随机的唯一标识符，在文件路径中使用，以确保每个上传的文件都有一个唯一的路径

// String flag = IdUtil.fastSimpleUUID();

// //构建上传文件的存储路径

// String rootFilePath = USER\_FILE\_PATH + flag + "-" + original;

// try {

// //使用 FileUtil 工具类的 writeBytes 方法将文件数据写入磁盘上的指定路径 rootFilePath

// FileUtil.writeBytes(file.getBytes(), rootFilePath);

// //刚刚上传的文件

// File image = new File(rootFilePath);

// if (image.length() >= 1024 \* 1024 / 10) {

// while (image.length() >= 1024 \* 1024 / 10) {

// //调用 ImageUtil 工具类的 scale 方法对图片进行缩放。缩放后的图片将覆盖原有文件，并且缩放比例为 2

// ImageUtil.scale(rootFilePath, rootFilePath,2,false);

// }

// }

// } catch (IOException e) {

// throw new MyException(EnumExceptionType.READ\_FILE\_ERROR);

// }

// return flag + "-" + original;

// }

//

//// /\*\*

//// \* 联系我们

//// \* @param contactRequest 联系我们请求

//// \* @return 是否成功

//// \* @throws MyException 通用异常

//// \*/

//// @Override

//// public Boolean contact(ContactRequest contactRequest) throws MyException{

//// try {

//// User user = userService.getUserById(sessionUtils.getUserId());

//// messageUtil.sendMail(sender, sender,"Bug report from:" + user.getUsername(),

//// contactRequest.getContent(), contactRequest.getImages(), jms);

//// } catch (Exception e) {

//// if (e instanceof MyException) {

//// throw (MyException)e;

//// }

//// e.printStackTrace();

//// throw new MyException(EnumExceptionType.SEND\_EMAIL\_FAILED);

//// }

////

//// return true;

//// }

//

//}

//

//

//

//

//

package fun.redamancyxun.chinese.backend.service.impl;

import fun.redamancyxun.chinese.backend.controller.lesson.response.LessonDetailResponse;

import fun.redamancyxun.chinese.backend.controller.lesson.response.LessonListResponse;

import fun.redamancyxun.chinese.backend.exception.EnumExceptionType;

import fun.redamancyxun.chinese.backend.exception.MyException;

import fun.redamancyxun.chinese.backend.service.LessonService;

import fun.redamancyxun.chinese.backend.util.ExcelReader;

import fun.redamancyxun.chinese.backend.util.SessionUtils;

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

import org.springframework.stereotype.Service;

/\*\*

\* 课程相关接口实现

\* @author Redamancy

\* @description 课程相关接口实现

\* @createDate 2024-10-16 22:39:04

\*/

@Service

public class LessonServiceImpl implements LessonService {

@Autowired

private SessionUtils sessionUtils;

/\*\*

\* 获取课程详情

\* @param unitId

\* @return LessonDetailResponse

\* @throws MyException

\*/

@Override

public LessonDetailResponse getLessonDetail(String unitId, Integer bookNumber) throws MyException {

sessionUtils.refreshData(null);

ExcelReader excelReader = ExcelReader.getInstance(bookNumber);

String text = excelReader.getDataMap().get(unitId);

if (text == null) {

throw new MyException(EnumExceptionType.TEXT\_NOT\_EXIST);

}

return LessonDetailResponse.builder()

.text(text)

.unitId(unitId)

.build();

}

/\*\*

\* 获取课程列表

\* @return LessonListResponse

\* @throws MyException

\*/

@Override

public LessonListResponse getLessonList(Integer bookNumber) throws MyException {

ExcelReader excelReader = ExcelReader.getInstance(bookNumber);

sessionUtils.refreshData(null);

return LessonListResponse.builder()

.lessons(excelReader.getCategoryCountMap())

.totalUnits(excelReader.getUnitCount())

.build();

}

}

package fun.redamancyxun.chinese.backend.service.impl;

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

import fun.redamancyxun.chinese.backend.entity.ScoreAction;

import fun.redamancyxun.chinese.backend.exception.EnumExceptionType;

import fun.redamancyxun.chinese.backend.exception.MyException;

import fun.redamancyxun.chinese.backend.mapper.ScoreActionMapper;

import fun.redamancyxun.chinese.backend.service.ScoreActionService;

import fun.redamancyxun.chinese.backend.service.UserService;

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

import org.springframework.stereotype.Service;

import java.util.List;

/\*\*

\* 评分操作相关接口实现类

\* @author Redamancy

\* @description 评分行为相关接口实现类

\* @createDate 2024-10-16 22:39:04

\*/

@Service

public class ScoreActionServiceImpl implements ScoreActionService {

@Autowired

private ScoreActionMapper scoreActionMapper;

@Autowired

private UserService userService;

/\*\*

\* 创建评分行为

\* @param userId 用户id

\* @param textId 文章id

\* @param speed 语速

\* @param pause 停顿

\* @param initialConsonants 声母

\* @param finalVowels 韵母

\* @param tones 声调

\* @param completeness 完整度

\* @param advice 评语&改进建议

\* @return 评分行为实体

\*/

@Override

public ScoreAction createScoreAction(String userId,String bookId, String textId, Integer speed, Integer pause, Integer initialConsonants, Integer finalVowels, Integer tones, Integer completeness, String advice) {

ScoreAction scoreAction = ScoreAction.builder()

.userId(userId)

.bookId(bookId)

.textId(textId)

.speed(speed)

.pause(pause)

.initialConsonants(initialConsonants)

.finalVowels(finalVowels)

.tones(tones)

.completeness(completeness)

.advice(advice)

.build();

if (scoreActionMapper.insert(scoreAction) == 0) {

throw new MyException(EnumExceptionType.INSERT\_FAILED);

}

return scoreAction;

}

/\*\*

\* 根据用户id查找评分行为

\* @param userId 用户id

\* @return 评分行为列表

\*/

@Override

public List<ScoreAction> findScoreActionByUserId(String userId) {

if (userId == null) {

throw new MyException(EnumExceptionType.DATA\_IS\_NULL);

}

if (userService.getUserById(userId) == null) {

throw new MyException(EnumExceptionType.USER\_NOT\_EXIST);

}

return scoreActionMapper.selectList(new QueryWrapper<ScoreAction>().eq("user\_id", userId));

}

}

package fun.redamancyxun.chinese.backend.service.impl;

import com.google.gson.Gson;

import com.google.gson.JsonObject;

import fun.redamancyxun.chinese.backend.common.CommonConstants;

import fun.redamancyxun.chinese.backend.controller.socre.request.AudioScoreRequest;

import fun.redamancyxun.chinese.backend.controller.socre.response.AudioScoreResponse;

import fun.redamancyxun.chinese.backend.entity.User;

import fun.redamancyxun.chinese.backend.exception.EnumExceptionType;

import fun.redamancyxun.chinese.backend.exception.MyException;

import fun.redamancyxun.chinese.backend.score.IseDemo;

import fun.redamancyxun.chinese.backend.service.ScoreActionService;

import fun.redamancyxun.chinese.backend.service.ScoreService;

import fun.redamancyxun.chinese.backend.service.UserService;

import fun.redamancyxun.chinese.backend.util.SessionUtils;

import okhttp3.\*;

import org.apache.commons.lang3.RandomStringUtils;

import org.apache.poi.ss.usermodel.Row;

import org.apache.poi.ss.usermodel.Sheet;

import org.apache.poi.ss.usermodel.Workbook;

import org.apache.poi.ss.usermodel.WorkbookFactory;

import org.apache.poi.xssf.usermodel.XSSFWorkbook;

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

import org.springframework.stereotype.Service;

import org.springframework.web.context.request.RequestAttributes;

import org.springframework.web.context.request.RequestContextHolder;

import java.io.\*;

import java.nio.charset.StandardCharsets;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.util.\*;

import java.util.concurrent.\*;

/\*\*

\* 评分相关接口实现类

\* @author Redamancy

\* @description 评分相关接口实现类

\* @createDate 2024-11-3 22:39:04

\*/

@Service

public class ScoreServiceImpl implements ScoreService {

@Autowired

private SessionUtils sessionUtils;

@Autowired

private ScoreActionService scoreActionService;

@Autowired

private UserService userService;

/\*\*

\* 将音频文件转换为pcm文件

\* @param data

\* @return pcm文件路径

\* @throws MyException

\*/

// TODO 16000Hz???

@Override

public String byte2pcm(byte[] data) throws MyException {

if (data == null) {

throw new MyException(EnumExceptionType.DATA\_IS\_NULL);

}

String filePath = CommonConstants.LINUX\_SCORE\_FILE\_PATH + "pcm/" + RandomStringUtils.randomAlphanumeric(12) + ".pcm";

// 使用 try-with-resources 确保 FileOutputStream 在使用后正确关闭

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

// 将 byte[] 数据写入文件

fos.write(data);

System.out.println("PCM 文件保存成功：" + filePath + "！");

} catch (IOException e) {

System.err.println("保存 PCM 文件时发生错误: " + e.getMessage());

throw new MyException(EnumExceptionType.IO\_ERROR, e.getMessage());

}

return filePath;

}

/\*\*

\* 利用pcm文件路径和文本内容生成音频-文本对照表

\* @param pcmPath

\* @param text

\* @return

\* @throws MyException

\*/

@Override

public void pcm2xlsx(String pcmPath, String text) throws MyException {

if (pcmPath == null || text == null) {

throw new MyException(EnumExceptionType.DATA\_IS\_NULL);

}

// 创建一个工作簿

Workbook workbook = new XSSFWorkbook();

// 创建一个工作表

Sheet sheet = workbook.createSheet("Audio-Text Mapping");

// 创建表头

Row headerRow = sheet.createRow(0);

headerRow.createCell(0).setCellValue("PCM文件路径");

headerRow.createCell(1).setCellValue("文本内容");

// 创建数据行

Row dataRow = sheet.createRow(1);

dataRow.createCell(0).setCellValue(pcmPath);

dataRow.createCell(1).setCellValue(text);

// 自动调整列宽

for (int i = 0; i < 2; i++) {

sheet.autoSizeColumn(i);

}

// 保存为Excel文件

try (FileOutputStream fileOut = new FileOutputStream(CommonConstants.LINUX\_SCORE\_FILE\_PATH + "AudioTextMapping.xlsx")) {

workbook.write(fileOut);

System.out.println("音频-文本对照表生成成功！");

} catch (IOException e) {

System.err.println("保存音频-文本对照表时发生错误: " + e.getMessage());

throw new MyException(EnumExceptionType.IO\_ERROR, e.getMessage());

}

// 关闭工作簿

try {

workbook.close();

} catch (IOException e) {

System.err.println("关闭工作簿时发生错误: " + e.getMessage());

throw new MyException(EnumExceptionType.IO\_ERROR, e.getMessage());

}

}

/\*\*

\* 对pcm音频文件利用讯飞模型量化，转换为xml文件

\*/

@Override

public void pcm2xml() throws Exception {

// try {

// // 定义 Python 解释器路径和脚本路径

// String pythonPath = CommonConstants.LINUX\_PYTHON\_PATH;

// String scriptPath = "src/main/java/fun/redamancyxun/chinese/backend/score/ise\_ws\_python3\_demo.py";

//

// /\*

// \* 创建 ProcessBuilder 并启动进程

// \* ProcessBuilder：用于创建和管理外部进程。这里传入了 Python 解释器路径和脚本路径作为参数

// \* pb.start()：启动外部进程，返回一个 Process 对象，用于与该进程进行交互

// \*/

// ProcessBuilder pb = new ProcessBuilder(pythonPath, scriptPath);

// Process proc = pb.start();

//

// /\*

// \* 读取标准输出流

// \* proc.getInputStream()：获取进程的标准输出流

// \* BufferedReader：用于读取输出流的每一行

// \* try-with-resources：确保 BufferedReader 在使用完毕后自动关闭（close()），避免资源泄漏

// \* while ((line = in.readLine()) != null)：逐行读取输出流的内容，并打印到控制台（建议使用日志框架替代）

// \*/

// try (BufferedReader in = new BufferedReader(new InputStreamReader(proc.getInputStream()))) {

// String line;

// while ((line = in.readLine()) != null) {

// System.out.println(line); // 使用日志框架替代

// }

// }

//

// /\*

// \* 读取错误流

// \* proc.getErrorStream()：获取进程的错误输出流

// \* BufferedReader：用于读取错误流的每一行

// \* while ((line = err.readLine()) != null)：逐行读取错误流的内容，并打印到控制台（建议使用日志框架替代）

// \*/

// try (BufferedReader err = new BufferedReader(new InputStreamReader(proc.getErrorStream()))) {

// String line;

// while ((line = err.readLine()) != null) {

// System.err.println(line); // 使用日志框架替代

// }

// }

//

// /\*

// \* 等待进程完成

// \* proc.waitFor(10, TimeUnit.SECONDS)：等待进程在 10 秒内完成。如果进程在 10 秒内完成，则返回 true；否则返回 false。

// \* proc.exitValue()：获取进程的退出码。如果退出码为 0，表示进程成功执行；否则表示进程执行失败。

// \* proc.destroyForcibly()：强制终止超时的进程。

// \* throw new MyException(...)：如果进程执行失败或超时，抛出自定义异常 MyException，并附带相应的错误信息

// \*/

// // TODO 考虑10s的时间是否足够

// if (proc.waitFor(10, TimeUnit.SECONDS)) {

// if (proc.exitValue() != 0) {

// throw new MyException(EnumExceptionType.PROCESS\_ERROR, "Python script execution failed with exit code: " + proc.exitValue());

// }

// } else {

// proc.destroyForcibly();

// throw new MyException(EnumExceptionType.PROCESS\_TIME\_OUT, "Python script execution timed out.");

// }

//

// System.out.println("讯飞模型量化成功！"); // 使用日志框架替代

//

// } catch (IOException e) {

// e.printStackTrace();

// throw new MyException(EnumExceptionType.IO\_ERROR, e.getMessage());

// } catch (InterruptedException e) {

// e.printStackTrace();

// throw new MyException(EnumExceptionType.INTERRUPTED\_ERROR, e.getMessage());

// }

Map<String, String> audioTextMapping = readAudioTextMapping(CommonConstants.LINUX\_SCORE\_FILE\_PATH + "AudioTextMapping.xlsx");

// 创建一个 CountDownLatch，用于等待所有线程完成

CountDownLatch latch = new CountDownLatch(audioTextMapping.size());

IseDemo.latch = latch;

for (Map.Entry<String, String> entry : audioTextMapping.entrySet()) {

try {

IseDemo.iseDemoScore(entry.getValue(), entry.getKey());

} catch (Exception e) {

e.printStackTrace();

throw e;

}

}

// 等待所有线程完成

latch.await();

// WebSocket 关闭后执行下一步操作

System.out.println("WebSocket closed, continuing with next steps...");

}

/\*\*

\* 将xml文件转换为文本文件

\* @return txt路径

\*/

@Override

public void xml2txt() {

Process proc1;

try {

String[] args1 = new String[]{CommonConstants.LINUX\_PYTHON\_PATH, CommonConstants.LINUX\_SCORE\_FILE\_PATH + "generate\_quantitative\_xlsx.py"};

System.out.println("Executing: " + Arrays.toString(args1));

proc1 = Runtime.getRuntime().exec(args1);

// 读取标准输出流

BufferedReader in = new BufferedReader(new InputStreamReader(proc1.getInputStream()));

String line = null;

while ((line = in.readLine()) != null) {

System.out.println(line);

}

in.close();

// 读取错误流

BufferedReader err1 = new BufferedReader(new InputStreamReader(proc1.getErrorStream()));

String errLine1 = null;

while ((errLine1 = err1.readLine()) != null) {

System.err.println(errLine1);

}

err1.close();

proc1.waitFor();

} catch (IOException e) {

e.printStackTrace();

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println("成功将xml文件转化为xlsx文件！");

Process proc2;

try {

String[] args2 = new String[]{CommonConstants.LINUX\_PYTHON\_PATH, CommonConstants.LINUX\_SCORE\_FILE\_PATH + "generate\_quantitative\_data.py"};

System.out.println("Executing: " + Arrays.toString(args2));

proc2 = Runtime.getRuntime().exec(args2);

// 读取标准输出流

BufferedReader in = new BufferedReader(new InputStreamReader(proc2.getInputStream()));

String line = null;

while ((line = in.readLine()) != null) {

System.out.println(line);

}

in.close();

// 读取错误流

BufferedReader err2 = new BufferedReader(new InputStreamReader(proc2.getErrorStream()));

String errLine2 = null;

while ((errLine2 = err2.readLine()) != null) {

System.err.println(errLine2);

}

err2.close();

proc2.waitFor();

} catch (IOException e) {

e.printStackTrace();

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println("成功将xlsx文件转化为txt文件！");

}

/\*\*

\* 音频文件评分

\* @param audioScoreRequest

\* @return AudioScoreResponse

\*/

@Override

public AudioScoreResponse score(AudioScoreRequest audioScoreRequest) throws Exception {

if (audioScoreRequest == null) {

throw new MyException(EnumExceptionType.DATA\_IS\_NULL);

}

// 获取当前用户的 userId

User user = userService.getUserById(sessionUtils.getUserId());

sessionUtils.refreshData(null);

// 音频文件、文章id、文章内容

byte[] audioFile;

try {

// 处理音频文件

audioFile = audioScoreRequest.getAudioFile().getBytes();

System.out.println("Received Audio Data: " + audioFile.length + " bytes");

} catch (Exception e) {

throw new MyException(EnumExceptionType.IO\_ERROR, e.getMessage());

}

String bookId = audioScoreRequest.getBookId();

String courseId = audioScoreRequest.getCourseId();

String courseContent = audioScoreRequest.getCourseContent();

// 将音频文件转化为pcm文件

String pcmPath = byte2pcm(audioFile);

// 将pcm文件转化为xlsx文件

pcm2xlsx(pcmPath, courseContent);

// 讯飞模型量化

pcm2xml();

// Thread.sleep(10000);

// 将XML数据保存到文件中

String xmlFilename = pcmPath.replace(".pcm", ".xml");

String xmlFilePath = xmlFilename.replace("pcm", "xml");

// 创建文件并确保目录存在

Path path = Paths.get(xmlFilePath);

Files.createDirectories(path.getParent()); // 确保目录存在

Files.write(path, IseDemo.xmlData.getBytes(StandardCharsets.UTF\_8)); // 直接写入内容

System.out.println("XML result saved as " + xmlFilePath);

// 将xml文件转化为txt文件

xml2txt();

String filePath = (getBaseName(pcmPath) + ".txt").replace("pcm", "txt");

String fileContent = readFileToString(filePath);

// 构建prompt

String system = "你是一名为留学生中文口语打分的国汉老师，请对学生的口语表现进行评估。你的评分标准基于以下六个维度：语速、停顿、声母、韵母、声调和完整度，每个维度给出0~5的整数评分。每个维度的评分标准和计算方式如下：" +

"{\\\"评分标准\\\":{\\\"准确度\\\":{\\\"声母/韵母\\\":{\\\"分类标准\\\":{\\\"正确率90%-100%\\\":{\\\"等级\\\":\\\"A\\\",\\\"量化值(g)\\\":5,\\\"描述\\\":\\\"发音准确，容易理解，接近母语水平，正确率很高\\\"," +

"\\\"计算方式\\\":\\\"1-(声韵错误数/总音节数)\\\"},\\\"正确率70%-90%\\\":{\\\"等级\\\":\\\"B\\\",\\\"量化值(g)\\\":3,\\\"描述\\\":\\\"基本可理解，仅有少量错误\\\",\\\"计算方式\\\":\\\"1-(声韵错误数/总音节数)\\\"}," +

"\\\"正确率0%-70%\\\":{\\\"等级\\\":\\\"C\\\",\\\"量化值(g)\\\":1,\\\"描述\\\":\\\"发音模糊，听懂困难\\\",\\\"计算方式\\\":\\\"1-(声韵错误数/总音节数)\\\"}}},\\\"声调\\\":{\\\"分类标准\\\":{\\\"正确率90%-100%\\\"" +

":{\\\"等级\\\":\\\"A\\\",\\\"量化值(g)\\\":5,\\\"描述\\\":\\\"声调准确，表达清晰\\\",\\\"计算方式\\\":\\\"调型错误数/总音节数\\\"},\\\"正确率70%-90%\\\":{\\\"等级\\\":\\\"B\\\",\\\"量化值(g)\\\":3,\\\"描述\\\":" +

"\\\"声调基本正确，偶有错误\\\",\\\"计算方式\\\":\\\"调型错误数/总音节数\\\"},\\\"正确率0%-70%\\\":{\\\"等级\\\":\\\"C\\\",\\\"量化值(g)\\\":1,\\\"描述\\\":\\\"声调错误频繁，难以理解\\\",\\\"计算方式\\\":\\\"调型错误数/总音节数\\\"}}}}," +

"\\\"流利度\\\":{\\\"语速\\\":{\\\"分类标准\\\":{\\\"每分钟120字及以上\\\":{\\\"等级\\\":\\\"A\\\",\\\"量化值(g)\\\":5,\\\"描述\\\":\\\"语言流畅，富有感情，基本没有认错的字，即使读错也能及时修改或巧妙避免重复式修改\\\",\\\"计算方式\\\":\\\"字数/时间（秒）\\\"}" +

",\\\"每分钟60-120字\\\":{\\\"等级\\\":\\\"B\\\",\\\"量化值(g)\\\":3,\\\"描述\\\":\\\"语言较为流畅，无意义的重复少、卡顿少，意识到错误能及时纠正\\\",\\\"计算方式\\\":\\\"字数/时间（秒）\\\"},\\\"每分钟60字以下\\\":{\\\"等级\\\":\\\"C\\\",\\\"量化值(g)\\\":1," +

"\\\"描述\\\":\\\"磕磕碰碰，无意义的重复多、卡顿多，意识到错误不能及时纠正\\\",\\\"计算方式\\\":\\\"字数/时间（秒）\\\"}}},\\\"停顿\\\":{\\\"分类标准\\\":{\\\"不合理停顿占比3%以内\\\":{\\\"等级\\\":\\\"A\\\",\\\"量化值(g)\\\":5,\\\"描述\\\":\\\"语言流畅，停顿自然，" +

"富有感情，基本没有认错的字，即使读错也能及时修改或巧妙避免重复式修改\\\",\\\"计算方式\\\":\\\"不合理停顿字数(fil)/总字数\\\"},\\\"不合理停顿3%-10%\\\":{\\\"等级\\\":\\\"B\\\",\\\"量化值(g)\\\":3,\\\"描述\\\":\\\"偶有停顿，整体较流畅，无意义的重复少、卡顿少，意识到错误能及" +

"时纠正\\\",\\\"计算方式\\\":\\\"不合理停顿字数(fil)/总字数\\\"},\\\"不合理停顿超过10%\\\":{\\\"等级\\\":\\\"C\\\",\\\"量化值(g)\\\":1,\\\"描述\\\":\\\"停顿频繁，影响理解，磕磕碰碰，无意义的重复多、卡顿多，意识到错误不能及时纠正\\\",\\\"计算方式\\\":\\\"不合理停顿字数(fil)" +

"/总字数\\\"}}}},\\\"完整度\\\":{\\\"完整性\\\":{\\\"分类标准\\\":{\\\"朗读完成度95%-100%\\\":{\\\"等级\\\":\\\"A\\\",\\\"量化值(g)\\\":5,\\\"描述\\\":\\\"内容完整，毫无缺漏\\\",\\\"计算方式\\\":\\\"1-((增读+漏读+回读)/总字数)\\\"},\\\"朗读完成度85%-94.9%\\\":{\\\"" +

"等级\\\":\\\"B\\\",\\\"量化值(g)\\\":3,\\\"描述\\\":\\\"有个别字词遗漏\\\",\\\"计算方式\\\":\\\"1-((增读+漏读+回读)/总字数)\\\"},\\\"朗读完成度84.9%及以下\\\":{\\\"等级\\\":\\\"C\\\",\\\"量化值(g)\\\":1,\\\"描述\\\":\\\"大段内容遗漏，理解困难\\\",\\\"计算方式\\\":\\\"" +

"1-((增读+漏读+回读)/总字数)\\\"}}}},\\\"感情\\\":{\\\"情感表达\\\":{\\\"分类标准\\\":{\\\"音节饱满，语调自然\\\":{\\\"等级\\\":\\\"A\\\",\\\"量化值(g)\\\":5,\\\"描述\\\":\\\"表现出色，情感丰富\\\"},\\\"表现平稳，缺少情感\\\":{\\\"等级\\\":\\\"B\\\",\\\"量化值(g)\\\":3,\\\"" +

"描述\\\":\\\"较为平淡，缺乏感染力\\\"},\\\"情感表达不佳，令人不适\\\":{\\\"等级\\\":\\\"C\\\",\\\"量化值(g)\\\":1,\\\"描述\\\":\\\"缺乏真诚，难以倾听\\\"}}}}}}";

String prompt = "我是一位学习中文口语的留学生，我的朗读内容是：{" + courseContent + "}。我的音频的量化数据是：{" + fileContent +

"}。你是一名为留学生中文口语打分的国汉老师，请根据system身份下给你输入的评分标准对我的口语表现进行评估，给我的语速、停顿、声母、韵母、声调和完整度五个维度" +

"分别给出0~5的整数评分并给出有实际意义和作用的评语。输出格式例子：语速：2；停顿：3；声母：3；韵母：3；声调：3；完整度：5；评语：朗读不够流利，有少许声母韵母错误，但完成度很好，希望继续努力~！";

System.out.println("PROMPT：" + prompt);

// TODO 调用训练好的模型API

// 调用模型API

// API Key

String apiKey = "5176fd66b5ebf072d3a4cb3cc7373e8b.GYxJkFGKPDaaMW3A";

// API endpoint

String url = "https://open.bigmodel.cn/api/paas/v4/chat/completions";

// Request body

JsonObject requestBody = new JsonObject();

requestBody.addProperty("model", "glm-4-9b:129142139:v1:rovpx6cx");

JsonObject systemMessage = new JsonObject();

systemMessage.addProperty("role", "system");

systemMessage.addProperty("content", system);

JsonObject userMessage = new JsonObject();

userMessage.addProperty("role", "user");

userMessage.addProperty("content", prompt);

requestBody.add("messages", new Gson().toJsonTree(new JsonObject[]{systemMessage, userMessage}));

// Create OkHttp client

OkHttpClient client = new OkHttpClient.Builder()

.connectTimeout(1000, TimeUnit.SECONDS) // 连接超时时间

.readTimeout(3000, TimeUnit.SECONDS) // 读取超时时间

.writeTimeout(1005, TimeUnit.SECONDS) // 写入超时时间

.build();;

// Create HTTP request

Request request = new Request.Builder()

.url(url)

.addHeader("Authorization", "Bearer " + apiKey)

.post(RequestBody.create(MediaType.parse("application/json"), requestBody.toString()))

.build();

// Execute request

String responseBody = null;

String responseResult = null;

try (Response response = client.newCall(request).execute()) {

if (response.isSuccessful()) {

if (response.body() != null) {

responseBody = response.body().string();

JsonObject jsonObject = new Gson().fromJson(responseBody, JsonObject.class);

responseResult = jsonObject.getAsJsonArray("choices").get(0).getAsJsonObject().get("message").getAsJsonObject().get("content").getAsString();

}

System.out.println("Response: " + responseBody);

System.out.println("Result: " + responseResult);

} else {

System.out.println("Request failed: " + response.code());

}

} catch (IOException e) {

throw new RuntimeException(e);

}

Integer speed = Integer.valueOf(responseResult.split("语速：")[1].split("；")[0].trim());

Integer pause = Integer.valueOf(responseResult.split("停顿：")[1].split("；")[0].trim());

Integer initialConsonants = Integer.valueOf(responseResult.split("声母：")[1].split("；")[0].trim());

Integer finalVowels = Integer.valueOf(responseResult.split("韵母：")[1].split("；")[0].trim());

Integer tones = Integer.valueOf(responseResult.split("声调：")[1].split("；")[0].trim());

Integer completeness = Integer.valueOf(responseResult.split("完整度：")[1].split("；")[0].trim());

String advice = responseResult.split("评语：")[1].trim();

// 创建评分记录

scoreActionService.createScoreAction(user.getId(), bookId, courseId, speed, pause, initialConsonants, finalVowels, tones, completeness, advice);

// 返回评分结果

AudioScoreResponse audioScoreResponse = AudioScoreResponse.builder()

.speed(speed)

.pause(pause)

.initialConsonants(initialConsonants)

.finalVowels(finalVowels)

.tones(tones)

.completeness(completeness)

.advice(advice)

.build();

System.out.println("评分完成！结果是：" + audioScoreResponse);

// String userId = user.getId();

//

// // 异步更新用户信息

// ExecutorService executorService = Executors.newFixedThreadPool(2);

//

// // 在异步任务开始前保存请求上下文

// RequestAttributes requestAttributes = RequestContextHolder.getRequestAttributes();

//

// // 异步任务

// executorService.submit(() -> {

// try {

//// // 在异步任务中恢复请求上下文

//// RequestContextHolder.setRequestAttributes(requestAttributes);

//

// userService.updateOneAdvice();

// System.out.println("异步任务完成: 更新用户建议");

// } catch (Exception e) {

// System.err.println("异步任务失败: " + e.getMessage());

// } finally {

//// // 确保清理请求上下文

//// RequestContextHolder.resetRequestAttributes();

// executorService.shutdown();

// }

// });

// userService.updateOneAdvice();

// // 异步更新用户信息

// CompletableFuture.runAsync(() -> {

// try {

// userService.updateOneAdvice();

// System.out.println("异步任务完成: 更新用户建议");

// } catch (Exception e) {

// System.err.println("异步任务失败: " + e.getMessage());

// }

// });

//

// // 计算程序用时

// long startTime = System.currentTimeMillis();

// userService.updateOneAdvice(user.getId());

// long endTime = System.currentTimeMillis();

// System.out.println("程序运行时间：" + (endTime - startTime) + "ms");

return audioScoreResponse;

}

// 读取文件

public static String readFileToString(String filePath) {

StringBuilder contentBuilder = new StringBuilder();

try (BufferedReader br = new BufferedReader(new FileReader(filePath))) {

String line;

while ((line = br.readLine()) != null) {

contentBuilder.append(line);

}

} catch (IOException e) {

e.printStackTrace();

}

return contentBuilder.toString();

}

// 获取文件名

public static String getBaseName(String filename) {

if (filename == null || filename.isEmpty()) {

throw new IllegalArgumentException("Filename cannot be null or empty.");

}

int dotIndex = filename.lastIndexOf('.');

if (dotIndex == -1) {

return filename; // 没有扩展名，直接返回文件名

} else {

return filename.substring(0, dotIndex); // 返回没有扩展名的部分

}

}

// 读取Excel文件获取音频-文本映射

private static Map<String, String> readAudioTextMapping(String excelFilePath) throws IOException {

Map<String, String> mapping = new HashMap<>();

FileInputStream fileInputStream = new FileInputStream(excelFilePath);

Workbook workbook = WorkbookFactory.create(fileInputStream);

Sheet sheet = workbook.getSheetAt(0);

for (Row row : sheet) {

if (row.getRowNum() == 0) continue; // 跳过标题行

String pcmPath = row.getCell(0).getStringCellValue();

String text = row.getCell(1).getStringCellValue();

mapping.put(pcmPath, text);

}

workbook.close();

fileInputStream.close();

return mapping;

}

}