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

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

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";

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 {

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"),

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"),

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"),

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 + '\'' +

'}';

}

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> {

@Builder.Default

private Integer pageNum = 1;

@Builder.Default

private Integer pageSize = 10;

private Long total;

private Integer pages;

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();

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

corsConfiguration.addAllowedHeader(CorsConfiguration.ALL);

corsConfiguration.addAllowedMethod(CorsConfiguration.ALL);

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 {

@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 {

@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();

}

}

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);

redisTemplate.setKeySerializer(redisKeySerializer);

redisTemplate.setHashKeySerializer(redisKeySerializer);

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 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。

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.port}")

private String port;

}

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;

@ApiModelProperty("受评文本id")

@TableField(value = "text\_id")

private String textId;

@ApiModelProperty("语速")

@TableField(value = "speed")

private Integer speed;

@ApiModelProperty("停顿")

@TableField(value = "pause")

private Integer pause;

@ApiModelProperty("声母")

@TableField(value = "initialConsonants")

private Integer initialConsonants;

@ApiModelProperty("韵母")

@TableField(value = "finalVowels")

private Integer finalVowels;

@ApiModelProperty("声调")

@TableField(value = "tones")

private Integer tones;

@ApiModelProperty("完整度")

@TableField(value = "completeness")

private Integer completeness;

@ApiModelProperty("评语&改进建议")

@TableField(value = "advice")

private String advice;

}

package fun.redamancyxun.chinese.backend.entity;

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 java.io.Serializable;

@ApiModel("user 用户信息")

@TableName(value ="user")

@Data

@NoArgsConstructor

@AllArgsConstructor

@Builder

public class User implements Serializable {

@ApiModelProperty("id")

@TableId(value = "id")

private String id;

@ApiModelProperty("密码")

@TableField(value = "password")

private String password;

@ApiModelProperty("用户名")

@TableField(value = "username")

private String username;

@ApiModelProperty("性别(0男，1女，2其他)")

@TableField(value = "gender")

private Integer gender;

@ApiModelProperty("年龄")

@TableField(value = "age")

private Integer age;

@ApiModelProperty("国籍")

@TableField(value = "nation")

private String nation;

@ApiModelProperty("手机号")

@TableField(value = "telephone")

private String telephone;

@ApiModelProperty("头像")

@TableField(value = "portrait")

private String portrait;

@ApiModelProperty("身份(0是学生，1是老师)")

@TableField(value = "role")

private Integer role;

@ApiModelProperty("发音特点、学习建议")

@TableField(value = "advice")

private String advice;

}

package fun.redamancyxun.chinese.backend.exception;

import lombok.AllArgsConstructor;

import lombok.Getter;

@Getter

@AllArgsConstructor

public enum EnumExceptionType {

UNKNOWN\_ERROR(999,"未知错误"),

SYSTEM\_INTERNAL\_ANOMALY(1000, "网络不给力，请稍后重试"),

UPDATE\_FAIL(1001,"上传失败"),

APPRAISE\_NOT\_EXIST(1002,"评价不存在"),

CREDIT\_NOT\_EXIST(1003,"评价明细不存在"),

INSERT\_FAILED(1004,"数据录入失败"),

TELEPHONE\_EXIST(1005,"手机号已存在"),

PASSWORD\_ERROR(1006,"密码错误"),

OLD\_PASSWORD\_ERROR(1006,"旧密码错误！"),

TEXT\_NOT\_EXIST(1007,"文本不存在"),

DATA\_IS\_NULL(1008,"传入数据不存在"),

IO\_ERROR(1009,"IO异常"),

INTERRUPTED\_ERROR(1010, "线程中断异常"),

PROCESS\_ERROR(1011, "进程错误"),

PROCESS\_TIME\_OUT(1012, "进程超时"),

ARTICLE\_NOT\_EXIST(3001,"帖子不存在"),

PARAMETER\_ERROR(3002,"参数错误"),

REDIS\_ERROR(3003, "Redis错误"),

NOTICE\_NOT\_EXIST(3004, "通知不存在"),

COMMENT\_NOT\_EXIST(3005, "评论不存在"),

STAR\_EXIST(3006, "收藏已存在"),

STAR\_FAILED(3007, "收藏失败"),

UNSTAR\_EXIT(3008, "未收藏"),

VIEW\_COUNT\_ERROR(3009, "浏览量错误"),

LIKE\_EXIST(3010, "已点赞"),

LIKE\_UNEXIST(3011, "未点赞"),

COMMENT\_NOT\_FOUND(3012, "评论不存在"),

REPLY\_NOT\_FOUND(3013, "回复不存在"),

CONSIGNEE\_NOT\_EXIST(3014, "收货地址不存在"),

EXPRESSORDER\_NOT\_EXIST(3015, "代取订单不存在"),

ACCEPTEDSSORDER\_NOT\_EXIST(3016, "接收的代取订单不存在"),

CANCEL\_FAIL(3017,"代取订单取消失败，原订单任务可能已完成或者已超时"),

RECEIVE\_FAIL(3018,"无法将代取订单设为已收到，原订单任务可能已取消或者已超时"),

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

USER\_NOT\_ADMIN(2002,"用户非管理员"),

NEED\_SESSION\_ID(2003,"未传入sessionId, 请传入会话id"),

LOGIN\_HAS\_OVERDUE(2004,"登录已过期"),

SESSION\_IS\_INVALID(2005,"该session数据库里没有, 请在header里key为session处对应有效的sessionId"),

SEND\_EMAIL\_FAILED(2024, "发送邮件失败, 请检查邮箱账号"),

EMAIL\_HAS\_BEEN\_SIGNED\_UP(2025, "该邮箱已被使用, 请更换邮箱"),

VERIFICATION\_CODE\_WRONG(2026, "邮箱验证码错误, 请输入正确的邮箱验证码"),

EMAIL\_NOT\_SIGNED\_UP(2027, "该邮箱尚未注册账号, 请输入正确的邮箱或先用此邮箱注册账号"),

PASSWORD\_NOT\_QUANTIFIED(2028, "密码强度不够, 请输入符合要求的密码"),

CAN\_NOT\_DELETE(2030, "超级管理员无法注销, 你这么牛逼, 注销自己干嘛"),

DO\_NOT\_SEND\_VERIFICATION\_CODE\_AGAIN(2031, "您上一次验证码尚未失效, 请勿重复发送验证码"),

VERIFICATION\_CODE\_HAS\_EXPIRED(2038, "验证码已过期, 请重新发送验证码"),

READ\_FILE\_ERROR(2039, "读取文件失败, 请检查文件"),

WX\_LOGIN\_BUSY(2040,"微信小程序繁忙，请稍候再试"),

WX\_LOGIN\_INVALID\_CODE(2041,"授权失败，请检查微信账号是否正常"),

WX\_LOGIN\_FREQUENCY\_REFUSED(2042,"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(2043,"微信授权未知错误,微信异常，请稍后再试"),

WX\_APPSECRET\_INVALID(2044,"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(2045,"请确保 grant\_type 字段值为 client\_credential,系统异常，请稍后再试"),

WX\_APPID\_INVALID(2046,"不合法的 AppID,系统异常，请稍后再试"),

WX\_CONTENT\_ILLEGAL(2047,"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(2048,"内容安全校验格式不合法,系统异常，请稍后再试"),

USER\_NOT\_EXIST(9001,"用户不存在"),

UPDATE\_FAILED(9007,"更新失败, 没有信息被更改"),

WRONG\_CODE(9021,"传入的code有误"),

AUTHEN\_NOT\_EXIST(9022,"认证不存在"),

STILL\_NOT\_VERIFIED(9023,"用户未身份认证"),

APPLY\_EXIST(9024,"已经发送过同样的申请啦！"),

USER\_NOT\_THIS\_TEAM(9025,"用户不是这个团队的！"),

SYSTEM\_ERROR(9026,"系统错误, 手机号获取失败"),

INPUT\_NULL(9027,"输入为空！"),

EMAIL\_PATTERN\_ERROR(9028,"邮箱地址有误"),

PERMISSION\_DENIED(9029, "您没有权限进行此操作"),

WECHAT\_ERROR(9030, "微信登录失败, 请检查微信登录"),

USER\_BANNED(9032, "用户被封禁, 用户被封禁"),

SESSION\_NOT\_FOUND(9041, "session不存在, 请清空wxStorage或重新登录！");

private final int errorCode;

private final String codeMessage;

}

package fun.redamancyxun.chinese.backend.exception;

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

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

import lombok.extern.slf4j.Slf4j;

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

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

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

import javax.servlet.http.HttpServletRequest;

import java.io.PrintWriter;

import java.io.StringWriter;

import java.util.Enumeration;

import java.util.HashMap;

import java.util.Map;

/\*\*

\* 统一异常处理

\* 线上日志

\*/

@Slf4j

@RestControllerAdvice

public class GlobalExceptionHandler {

@Autowired

private SessionUtils sessionUtil;

@ExceptionHandler(value = Exception.class)

public Result defaultErrorHandler(HttpServletRequest request, Exception e) {

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

Enumeration<?> temp = request.getParameterNames();

if (null != temp) {

while (temp.hasMoreElements()) {

String en = (String) temp.nextElement();

String value = request.getParameter(en);

res.put(en, value);

}

}

log.error("args: " + res.toString());

log.error("e:" + e.getMessage());

StringWriter sw = new StringWriter();

PrintWriter pw = new PrintWriter(sw, true);

e.printStackTrace(pw);

pw.flush();

sw.flush();

log.error("exception: " + sw.toString());

if (e instanceof MyException) {

log.error("MyException: " + ((MyException) e).getEnumExceptionType().getCodeMessage() + "\n" + sw.toString());

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

}

return Result.fail(sw.toString(), null);

}

}

package fun.redamancyxun.chinese.backend.exception;

import lombok.Getter;

@Getter

public class MyException extends RuntimeException {

private EnumExceptionType enumExceptionType;

private Object errorMsg;

public MyException(){

this.enumExceptionType = EnumExceptionType.UNKNOWN\_ERROR;

}

public MyException(EnumExceptionType enumExceptionType) {

this.enumExceptionType = enumExceptionType;

}

public MyException(EnumExceptionType commonErrorCode, Object errorMsg) {

this.enumExceptionType = commonErrorCode;

this.errorMsg = errorMsg;

}

}

package fun.redamancyxun.chinese.backend.mapper;

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

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

import org.apache.ibatis.annotations.Mapper;

@Mapper

public interface AdviceActionMapper extends BaseMapper<AdviceAction> {

}

package fun.redamancyxun.chinese.backend.mapper;

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

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

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

import org.apache.ibatis.annotations.Mapper;

import org.apache.ibatis.annotations.Select;

import java.util.List;

/\*\*

\* @description 针对表【score\_action(练习与评价记录)】的数据库操作Mapper

\*/

@Mapper

public interface ScoreActionMapper extends BaseMapper<ScoreAction> {

/\*\*

\* 查询历史练习与评价记录

\* @param userId 用户id

\* @return List<ScoreAction>

\*/

@Select("SELECT \* FROM score\_action WHERE user\_id = #{userId}")

List<ScoreAction> selectList(String userId);

}

package fun.redamancyxun.chinese.backend.mapper;

import com.baomidou.mybatisplus.core.conditions.Wrapper;

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

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

import org.apache.ibatis.annotations.Mapper;

import org.apache.ibatis.annotations.Param;

import org.apache.ibatis.annotations.Select;

import java.util.List;

/\*\*

\* @description 针对表【user(用户信息)】的数据库操作Mapper

\*/

@Mapper

public interface UserMapper extends BaseMapper<User> {

@Select("SELECT \* FROM user WHERE id =#{id}")

User getUserById(String id);

@Select("SELECT id FROM user;")

List<String> selectAllUserId();

}

package fun.redamancyxun.chinese.backend.score;

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

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

import java.io.\*;

import java.util.\*;

public class GenerateQuantitativeData {

public static void main(String[] args) {

String folderPath = "src/main/java/fun/redamancyxun/chinese/backend/score/xlsx/";

String savePath = "src/main/java/fun/redamancyxun/chinese/backend/score/txt/";

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

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

Workbook workbook = new XSSFWorkbook(file)) {

Sheet sheet = workbook.getSheetAt(0);

Iterator<Row> rowIterator = sheet.iterator();

if (rowIterator.hasNext()) {

rowIterator.next();

}

while (rowIterator.hasNext()) {

Row row = rowIterator.next();

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

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

audioTextMapping.put(pcmPath, text);

}

} catch (IOException e) {

e.printStackTrace();

}

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

String fileFullName = fileName.substring(0, fileName.lastIndexOf('.')) + ".xlsx";

File file = new File(folderPath + fileFullName);

try (FileInputStream excelFile = new FileInputStream(file);

Workbook workbook = new XSSFWorkbook(excelFile)) {

Sheet sheet = workbook.getSheetAt(0);

Row headerRow = sheet.getRow(0);

double smoothnessScore = headerRow.getCell(4).getNumericCellValue();

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

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

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

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

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

File outputFile = new File(savePath + outputFileName);

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

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);

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 {

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<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);

}

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

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

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

File xmlFile = new File(xmlFilename);

DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();

Document doc = dBuilder.parse(xmlFile);

doc.getDocumentElement().normalize();

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);

}

}

}

}

}

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

\*/

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());

}

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;

}

@Override

public void onOpen(WebSocket webSocket, Response response) {

super.onOpen(webSocket, response);

new Thread(() -> {

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

ssb(webSocket);

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

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

byte[] buffer = new byte[frameSize];

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();

}

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")

.add("group", "adult")

/\*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("sub",sub)

.add("ent",ent)

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

.add("cmd", "ssb")

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

.add("aue", "raw")

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

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

.add("status", 0)

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

webSocket.send(p.toString());

}

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")

);

webSocket.send(p.toString());

}

@Override

public void onMessage(WebSocket webSocket, String text) {

super.onMessage(webSocket, 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) {

}

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"));

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

} catch (Exception e) {

e.printStackTrace();

}

webSocket.close(1000, "");

latch.countDown();

} else {

}

}

}

}

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());

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");

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);

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

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();

}

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;

}

}

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 {

AdviceAction createAdviceAction(String userId, String advice);

List<AdviceAction> findAdviceActionByUserId(String userId);

}

package fun.redamancyxun.chinese.backend.service;

import org.springframework.web.multipart.MultipartFile;

/\*\*

\* @author Redamancy

\* @description 全局服务

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

\*/

public interface GlobalService {

String uploadImage(MultipartFile file);

}

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 {

LessonDetailResponse getLessonDetail(String unitId, Integer bookNumber);

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 {

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

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 {

String byte2pcm(byte[] data);

void pcm2xlsx(String pcmPath, String text);

void pcm2xml() throws Exception;

void xml2txt();

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> {

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

Integer checkLogin() throws MyException;

UserInfoResponse logout();

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

UserInfoResponse getUserInfo();

UserInfoResponse updateUserInfo(UpdateUserInfoRequest updateUserInfoRequest);

UserInfoResponse updateOneAdvice(String userId);

UserInfoResponse updateOneAdvice();

UserInfoResponse updateAllAdvice();

UserInfoResponse changePassword(String oldPassword, String newPassword);

String uploadPortrait(MultipartFile file);

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 {

User user = userService.getUserById(userId);

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 {

User user = userService.getUserById(userId);

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

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

return adviceActionMapper.selectList(queryWrapper);

}

}

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

\*/

@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";

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

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);

}

Workbook workbook = new XSSFWorkbook();

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

Row headerRow = sheet.createRow(0);

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

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

Row dataRow = sheet.createRow(1);

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

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

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

sheet.autoSizeColumn(i);

}

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());

}

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 {

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

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;

}

}

latch.await();

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);

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

String line = null;

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

System.out.println(line);

}

in.close();

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);

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

String line = null;

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

System.out.println(line);

}

in.close();

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);

}

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

sessionUtils.refreshData(null);

byte[] audioFile;

try {

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();

String pcmPath = byte2pcm(audioFile);

pcm2xlsx(pcmPath, courseContent);

pcm2xml();

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

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

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);

xml2txt();

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

String fileContent = readFileToString(filePath);

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);

String apiKey = "5176fd66b5ebf072d3a4cb3cc7373e8b.GYxJkFGKPDaaMW3A";

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

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}));

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();;

Request request = new Request.Builder()

.url(url)

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

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

.build();

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();

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

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);

return audioScoreResponse;

}

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();

}

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

}

}

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;

}

}

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

import cn.hutool.core.io.FileUtil;

import cn.hutool.core.util.IdUtil;

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

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

import com.google.gson.Gson;

import com.google.gson.JsonObject;

import fun.redamancyxun.chinese.backend.config.YmlConfig;

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

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

import fun.redamancyxun.chinese.backend.dto.SessionData;

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 fun.redamancyxun.chinese.backend.mapper.UserMapper;

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

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

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

import fun.redamancyxun.chinese.backend.util.\*;

import okhttp3.\*;

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

import org.springframework.scheduling.annotation.Async;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import org.springframework.web.multipart.MultipartFile;

import javax.servlet.http.HttpServletRequest;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import java.util.UUID;

import java.util.concurrent.TimeUnit;

import static fun.redamancyxun.chinese.backend.common.CommonConstants.\*;

/\*\*

\* @description 针对表【user(用户信息)】的数据库操作Service实现

\*/

@Service

public class UserServiceImpl extends ServiceImpl<UserMapper, User> implements UserService{

@Autowired

SessionUtils sessionUtils;

@Autowired

UserMapper userMapper;

@Autowired

ScoreActionService scoreActionService;

@Autowired

AdviceActionService adviceActionService;

@Autowired

YmlConfig ymlConfig;

@Autowired

private RedisUtils redisUtils;

@Autowired

private HttpServletRequest request;

/\*\*

\* 注册

\* @return UserInfoResponse

\* @throws MyException 通用异常

\*/

@Override

@Transactional(rollbackFor = MyException.class)

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

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

queryWrapper.eq("telephone", telephone);

if (userMapper.selectCount(queryWrapper) != 0){

throw new MyException(EnumExceptionType.TELEPHONE\_EXIST);

}

User user = User.builder()

.id(UUID.randomUUID().toString().substring(0, 8))

.role(role)

.username("小吉祥" + UUID.randomUUID().toString().substring(0, 4))

.portrait(IMAGE\_PATH + "defaultAvatar.svg")

.gender(2)

.advice("在中文的世界里，每一个字都是一颗星星，每一次发音都是一次星际旅行。虽然旅程漫长，但每一步的探索都是通向光辉的过程。继续前行，书写属于你的星辰大海。")

.age(0)

.nation("吉祥国")

.password(password)

.telephone(telephone)

.build();

if (userMapper.insert(user) == 0){

throw new MyException(EnumExceptionType.INSERT\_FAILED);

}

String sessionId = sessionUtils.generateSessionId();

SessionData sessionData = new SessionData(user);

sessionUtils.setSessionId(sessionId);

redisUtils.set(user.getId(), sessionId, 86400);

redisUtils.set(sessionId, sessionData, 86400);

return new UserInfoResponse(user, sessionId);

}

/\*\*

\* 登录

\* @param telephone 手机号

\* @param password 密码

\* @param role 角色

\* @return LoginInfo

\* @throws Exception 异常

\*/

@Override

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

QueryWrapper<User> userQueryWrapper = new QueryWrapper<>();

userQueryWrapper.eq("telephone", telephone);

userQueryWrapper.eq("role", role);

User user = userMapper.selectOne(userQueryWrapper);

if (user == null){

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

}

if (!user.getPassword().equals(password)){

throw new MyException(EnumExceptionType.PASSWORD\_ERROR);

}

String sessionId = sessionUtils.generateSessionId();

SessionData sessionData = new SessionData(user);

sessionUtils.setSessionId(sessionId);

redisUtils.set(user.getId(), sessionId, 86400);

redisUtils.set(sessionId, sessionData, 86400);

return new UserInfoResponse(user, sessionId);

}

/\*\*

\* 根据cookie检测登录状况

\* @return -1 会话过期，0 表示未登录，1 表示学生，2 表示老师，3 表示管理员，4 表示超级管理员

\* @throws MyException 通用异常

\*/

@Override

public Integer checkLogin() throws MyException {

String key = request.getHeader("session");

if (key == null || !redisUtils.hasKey(key)) {

return 0;

}

if (redisUtils.isExpire(key)) {

redisUtils.del(key);

return -1;

}

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

sessionUtils.refreshData(null);

if (user.getRole() == 0) {

return 1;

}

if (user.getRole() == 1) {

return 2;

}

return 4;

}

/\*\*

\* 注销

\*

\* @return UserInfoResponse

\*/

@Override

public UserInfoResponse logout() {

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

sessionUtils.invalidate();

return new UserInfoResponse(user);

}

/\*\*

\* 获取用户信息

\* @return UserInfoResponse

\* @throws MyException 通用异常

\*/

@Override

public UserInfoResponse getUserInfo() {

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

sessionUtils.refreshData(null);

QueryWrapper<ScoreAction> scoreActionQueryWrapper = new QueryWrapper<>();

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

List<ScoreAction> history = scoreActionService.findScoreActionByUserId(user.getId());

return new UserInfoResponse(user, history);

}

/\*\*

\* 更新用户信息

\* @param userSignupRequest 用户信息

\* @return UserInfoResponse

\* @throws MyException 通用异常

\*/

@Override

public UserInfoResponse updateUserInfo(UpdateUserInfoRequest userSignupRequest) {

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

sessionUtils.refreshData(null);

if (userSignupRequest.getAge() != null) {

user.setAge(userSignupRequest.getAge());

}

if (userSignupRequest.getGender() != null) {

user.setGender(userSignupRequest.getGender());

}

if (userSignupRequest.getNation() != null) {

user.setNation(userSignupRequest.getNation());

}

if (userSignupRequest.getUsername() != null) {

user.setUsername(userSignupRequest.getUsername());

}

if (userMapper.updateById(user) == 0){

throw new MyException(EnumExceptionType.UPDATE\_FAILED);

}

return new UserInfoResponse(user);

}

/\*\*

\* 修改密码

\* @param oldPassword 旧密码

\* @param newPassword 新密码

\* @return UserInfoResponse

\* @throws MyException 通用异常

\*/

@Override

public UserInfoResponse changePassword(String oldPassword, String newPassword) throws MyException {

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

sessionUtils.refreshData(null);

if (!user.getPassword().equals(oldPassword)){

throw new MyException(EnumExceptionType.OLD\_PASSWORD\_ERROR);

}

user.setPassword(newPassword);

if (userMapper.updateById(user) == 0){

throw new MyException(EnumExceptionType.UPDATE\_FAILED);

}

return new UserInfoResponse(user);

}

/\*\*

\* 上传头像

\* @param file 头像文件

\* @return String

\* @throws MyException 通用异常

\*/

@Override

public String uploadPortrait(MultipartFile file) throws MyException {

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

sessionUtils.refreshData(null);

if (user == null) {

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

}

String originalFilename = file.getOriginalFilename();

String flag = IdUtil.fastSimpleUUID();

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

try {

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

} catch (IOException e) {

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

}

String link = IMAGE\_PATH + flag + "-" + originalFilename;

user.setPortrait(link);

if (userMapper.updateById(user) == 0) {

throw new MyException(EnumExceptionType.UPDATE\_FAIL);

}

return link;

}

/\*\*

\* 根据userId获取用户信息

\* @param userId 用户id

\* @return User

\* @throws MyException 通用异常

\*/

@Override

public User getUserById(String userId) throws MyException{

User user = userMapper.selectById(userId);

sessionUtils.refreshData(null);

if (user == null){

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

}

return user;

}

/\*\*

\* 更新一个用户的“发育特点、学习建议”

\* @param userId 用户id

\* @throws MyException 通用异常

\*/

@Override

public UserInfoResponse updateOneAdvice(String userId) throws MyException {

User user = getUserById(userId);

List<ScoreAction> history = scoreActionService.findScoreActionByUserId(userId);

if (history.isEmpty()) {

return new UserInfoResponse(user);

}

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

for (int i = history.size() - 1; i > 0; i--) {

if (scoreList.size() < 10) {

String json = "{\"语速\": \"" + history.get(i).getSpeed() + "\",\"停顿\": \"" + history.get(i).getPause() + "\",\"声母\": \"" + history.get(i).getInitialConsonants() + "\",\"韵母\": \"" + history.get(i).getFinalVowels() + "\",\"声调\": \"" + history.get(i).getTones() + "\",\"完整度\": \"" + history.get(i).getCompleteness() + "\",\"评语&更改建议\": \"" + history.get(i).getAdvice() + "\"}";

scoreList.add(json);

}

}

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

exampleScoreList.add("{\"语速\": \"" + 5 + "\",\"停顿\": \"" + 4 + "\",\"声母\": \"" + 3 + "\",\"韵母\": \"" + 3 + "\",\"声调\": \"" + 3 + "\",\"完整度\": \"" + 5 + "\",\"评语&更改建议\": \"" + "朗读流利，声母韵母较为标准，有自行纠错意识，声调略有不足，朗读完成度高，表现出色！" + "\"}");

exampleScoreList.add("{\"语速\": \"" + 4 + "\",\"停顿\": \"" + 4 + "\",\"声母\": \"" + 4 + "\",\"韵母\": \"" + 3 + "\",\"声调\": \"" + 4 + "\",\"完整度\": \"" + 5 + "\",\"评语&更改建议\": \"" + "朗读流利，声母韵母较为标准，有自行纠错意识，声调略有进步，出现了误读字。完成度很好，表现出色！" + "\"}");

exampleScoreList.add("{\"语速\": \"" + 4 + "\",\"停顿\": \"" + 3 + "\",\"声母\": \"" + 4 + "\",\"韵母\": \"" + 3 + "\",\"声调\": \"" + 3 + "\",\"完整度\": \"" + 5 + "\",\"评语&更改建议\": \"" + "朗读较为流利，声母韵母较为标准，有自行纠错意识，声调略有进步，儿化音发音不够自然。完成度很好，继续努力！" + "\"}");

exampleScoreList.add("{\"语速\": \"" + 4 + "\",\"停顿\": \"" + 3 + "\",\"声母\": \"" + 3 + "\",\"韵母\": \"" + 4 + "\",\"声调\": \"" + 3 + "\",\"完整度\": \"" + 5 + "\",\"评语&更改建议\": \"" + "朗读较为流利，声母韵母较为标准，有自行纠错意识，声调略有不足，完成度很好，继续努力！" + "\"}");

exampleScoreList.add("{\"语速\": \"" + 5 + "\",\"停顿\": \"" + 3 + "\",\"声母\": \"" + 3 + "\",\"韵母\": \"" + 4 + "\",\"声调\": \"" + 3 + "\",\"完整度\": \"" + 5 + "\",\"评语&更改建议\": \"" + "朗读较为流利，声母韵母较为标准，有自行纠错意识，声调略有不足，完成度很好，表现出色！" + "\"}");

exampleScoreList.add("{\"语速\": \"" + 4 + "\",\"停顿\": \"" + 3 + "\",\"声母\": \"" + 3 + "\",\"韵母\": \"" + 4 + "\",\"声调\": \"" + 3 + "\",\"完整度\": \"" + 5 + "\",\"评语&更改建议\": \"" + "朗读流畅度不足，声母韵母较为标准，有自行纠错意识，声调略有不足，儿化音发音不够自然，完成度很好，继续努力！" + "\"}");

exampleScoreList.add("{\"语速\": \"" + 5 + "\",\"停顿\": \"" + 4 + "\",\"声母\": \"" + 4 + "\",\"韵母\": \"" + 3 + "\",\"声调\": \"" + 3 + "\",\"完整度\": \"" + 5 + "\",\"评语&更改建议\": \"" + "朗读较为流利，声母韵母发音较标准，有自行纠错意识，声调略有不足，完成度很好，表现出色！" + "\"}");

exampleScoreList.add("{\"语速\": \"" + 5 + "\",\"停顿\": \"" + 4 + "\",\"声母\": \"" + 4 + "\",\"韵母\": \"" + 4 + "\",\"声调\": \"" + 3 + "\",\"完整度\": \"" + 5 + "\",\"评语&更改建议\": \"" + "朗读较为流利，声母韵母较为标准，有自行纠错意识，儿化音发音较为自然。完成度很好，表现出色！" + "\"}");

exampleScoreList.add("{\"语速\": \"" + 4 + "\",\"停顿\": \"" + 4 + "\",\"声母\": \"" + 3 + "\",\"韵母\": \"" + 3 + "\",\"声调\": \"" + 2 + "\",\"完整度\": \"" + 5 + "\",\"评语&更改建议\": \"" + "朗读较为流利，声母韵母较为标准，有自行纠错意识，声调略有不足，有误读字，朗读完成度高，表现出色！" + "\"}");

exampleScoreList.add("{\"语速\": \"" + 5 + "\",\"停顿\": \"" + 5 + "\",\"声母\": \"" + 3 + "\",\"韵母\": \"" + 4 + "\",\"声调\": \"" + 3 + "\",\"完整度\": \"" + 5 + "\",\"评语&更改建议\": \"" + "朗读非常流利，声母韵母较为标准，声调基本准确，完成度高，表现出色！" + "\"}");

String exampleAdvice = "发音特点：整体来说朗读较为流利，完成度高。声母上存在n、l不分，r发音含混的问题，韵母发音情况较声母来说更好，儿化音有时发音不够自然。声调部分问题较为明显，存在四声调混淆的问题，需要多加注意与练习。整体朗读有节奏感，基本知道应该在何处停顿，但仍有对课文不够熟悉所以断句不恰当的情况，且朗读过程中有自行纠错意识，值得肯定。\n学习建议：可以多听课文的标准录音进行声韵调的模仿学习，在具体声母发音上可以询问任课教师，结合发音理论和实际教学经验来学会运用正确的发生部位发音。还可以多和中国语伴进行语言交流，听一听他们是如何断句的。需要多多复习所学，巩固朗读知识。";

String system = "你是一名经验丰富的对外汉语教师，专门负责根据留学生的近期中文口语成绩提供个性化的分析和建议。请根据留学生近期的中文口语成绩，深入分析学生的发音特点，并针对性地提出有效的学习建议。这些建议应考虑到学生的个体差异，既能帮助他们提升口语能力，又具有实际的参考价值和实现可能性。学生的口语成绩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，我近期中文口语成绩是：{" + scoreList + "}。口语成绩是基于以下六个维度：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，并且有对应的评语&更改建议，在system身份下说明了具体的评分标准。你是一名经验丰富的对外汉语教师，专门负责根据留学生的近期中文口语成绩提供个性化的分析和建议。请根据我近期的中文口语成绩，深入分析我的发音特点，并针对性地提出有效的学习建议。这些建议应考虑到学生的个体差异，既能帮助我们提升口语能力，又具有实际的参考价值和实现可能性。示例：比如有一位留学生近期的中文口语成绩为：{" + exampleScoreList + "}，你的输出示例是：{" + exampleAdvice + "}。";

System.out.println(prompt);

String apiKey = "5176fd66b5ebf072d3a4cb3cc7373e8b.GYxJkFGKPDaaMW3A";

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

JsonObject requestBody = new JsonObject();

requestBody.addProperty("model", "glm-4-flash");

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}));

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();;

Request request = new Request.Builder()

.url(url)

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

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

.build();

String responseBody = null;

String responseAdvice = 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);

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

}

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

System.out.println("Advice: " + responseAdvice);

} else {

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

}

} catch (IOException e) {

throw new RuntimeException(e);

}

adviceActionService.createAdviceAction(userId, responseAdvice);

user.setAdvice(responseAdvice);

if (userMapper.updateById(user) == 0) {

throw new MyException(EnumExceptionType.UPDATE\_FAILED);

}

return new UserInfoResponse(user);

}

/\*\*

\* 更新当前用户的“发育特点、学习建议”

\* @throws MyException 通用异常

\*/

@Override

public UserInfoResponse updateOneAdvice() throws MyException {

String userId = sessionUtils.getUserId();

User user = getUserById(userId);

List<ScoreAction> history = scoreActionService.findScoreActionByUserId(userId);

if (history.isEmpty()) {

return new UserInfoResponse(user);

}

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

for (int i = history.size() - 1; i > 0; i--) {

if (scoreList.size() < 10) {

String json = "{\"语速\": \"" + history.get(i).getSpeed() + "\",\"停顿\": \"" + history.get(i).getPause() + "\",\"声母\": \"" + history.get(i).getInitialConsonants() + "\",\"韵母\": \"" + history.get(i).getFinalVowels() + "\",\"声调\": \"" + history.get(i).getTones() + "\",\"完整度\": \"" + history.get(i).getCompleteness() + "\",\"评语&更改建议\": \"" + history.get(i).getAdvice() + "\"}";

scoreList.add(json);

}

}

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

exampleScoreList.add("{\"语速\": \"" + 5 + "\",\"停顿\": \"" + 4 + "\",\"声母\": \"" + 3 + "\",\"韵母\": \"" + 3 + "\",\"声调\": \"" + 3 + "\",\"完整度\": \"" + 5 + "\",\"评语&更改建议\": \"" + "朗读流利，声母韵母较为标准，有自行纠错意识，声调略有不足，朗读完成度高，表现出色！" + "\"}");

exampleScoreList.add("{\"语速\": \"" + 4 + "\",\"停顿\": \"" + 4 + "\",\"声母\": \"" + 4 + "\",\"韵母\": \"" + 3 + "\",\"声调\": \"" + 4 + "\",\"完整度\": \"" + 5 + "\",\"评语&更改建议\": \"" + "朗读流利，声母韵母较为标准，有自行纠错意识，声调略有进步，出现了误读字。完成度很好，表现出色！" + "\"}");

exampleScoreList.add("{\"语速\": \"" + 4 + "\",\"停顿\": \"" + 3 + "\",\"声母\": \"" + 4 + "\",\"韵母\": \"" + 3 + "\",\"声调\": \"" + 3 + "\",\"完整度\": \"" + 5 + "\",\"评语&更改建议\": \"" + "朗读较为流利，声母韵母较为标准，有自行纠错意识，声调略有进步，儿化音发音不够自然。完成度很好，继续努力！" + "\"}");

exampleScoreList.add("{\"语速\": \"" + 4 + "\",\"停顿\": \"" + 3 + "\",\"声母\": \"" + 3 + "\",\"韵母\": \"" + 4 + "\",\"声调\": \"" + 3 + "\",\"完整度\": \"" + 5 + "\",\"评语&更改建议\": \"" + "朗读较为流利，声母韵母较为标准，有自行纠错意识，声调略有不足，完成度很好，继续努力！" + "\"}");

exampleScoreList.add("{\"语速\": \"" + 5 + "\",\"停顿\": \"" + 3 + "\",\"声母\": \"" + 3 + "\",\"韵母\": \"" + 4 + "\",\"声调\": \"" + 3 + "\",\"完整度\": \"" + 5 + "\",\"评语&更改建议\": \"" + "朗读较为流利，声母韵母较为标准，有自行纠错意识，声调略有不足，完成度很好，表现出色！" + "\"}");

exampleScoreList.add("{\"语速\": \"" + 4 + "\",\"停顿\": \"" + 3 + "\",\"声母\": \"" + 3 + "\",\"韵母\": \"" + 4 + "\",\"声调\": \"" + 3 + "\",\"完整度\": \"" + 5 + "\",\"评语&更改建议\": \"" + "朗读流畅度不足，声母韵母较为标准，有自行纠错意识，声调略有不足，儿化音发音不够自然，完成度很好，继续努力！" + "\"}");

exampleScoreList.add("{\"语速\": \"" + 5 + "\",\"停顿\": \"" + 4 + "\",\"声母\": \"" + 4 + "\",\"韵母\": \"" + 3 + "\",\"声调\": \"" + 3 + "\",\"完整度\": \"" + 5 + "\",\"评语&更改建议\": \"" + "朗读较为流利，声母韵母发音较标准，有自行纠错意识，声调略有不足，完成度很好，表现出色！" + "\"}");

exampleScoreList.add("{\"语速\": \"" + 5 + "\",\"停顿\": \"" + 4 + "\",\"声母\": \"" + 4 + "\",\"韵母\": \"" + 4 + "\",\"声调\": \"" + 3 + "\",\"完整度\": \"" + 5 + "\",\"评语&更改建议\": \"" + "朗读较为流利，声母韵母较为标准，有自行纠错意识，儿化音发音较为自然。完成度很好，表现出色！" + "\"}");

exampleScoreList.add("{\"语速\": \"" + 4 + "\",\"停顿\": \"" + 4 + "\",\"声母\": \"" + 3 + "\",\"韵母\": \"" + 3 + "\",\"声调\": \"" + 2 + "\",\"完整度\": \"" + 5 + "\",\"评语&更改建议\": \"" + "朗读较为流利，声母韵母较为标准，有自行纠错意识，声调略有不足，有误读字，朗读完成度高，表现出色！" + "\"}");

exampleScoreList.add("{\"语速\": \"" + 5 + "\",\"停顿\": \"" + 5 + "\",\"声母\": \"" + 3 + "\",\"韵母\": \"" + 4 + "\",\"声调\": \"" + 3 + "\",\"完整度\": \"" + 5 + "\",\"评语&更改建议\": \"" + "朗读非常流利，声母韵母较为标准，声调基本准确，完成度高，表现出色！" + "\"}");

String exampleAdvice = "发音特点：整体来说朗读较为流利，完成度高。声母上存在n、l不分，r发音含混的问题，韵母发音情况较声母来说更好，儿化音有时发音不够自然。声调部分问题较为明显，存在四声调混淆的问题，需要多加注意与练习。整体朗读有节奏感，基本知道应该在何处停顿，但仍有对课文不够熟悉所以断句不恰当的情况，且朗读过程中有自行纠错意识，值得肯定。\n学习建议：可以多听课文的标准录音进行声韵调的模仿学习，在具体声母发音上可以询问任课教师，结合发音理论和实际教学经验来学会运用正确的发生部位发音。还可以多和中国语伴进行语言交流，听一听他们是如何断句的。需要多多复习所学，巩固朗读知识。";

String system = "你是一名经验丰富的对外汉语教师，专门负责根据留学生的近期中文口语成绩提供个性化的分析和建议。请根据留学生近期的中文口语成绩，深入分析学生的发音特点，并针对性地提出有效的学习建议。这些建议应考虑到学生的个体差异，既能帮助他们提升口语能力，又具有实际的参考价值和实现可能性。学生的口语成绩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，我近期中文口语成绩是：{" + scoreList + "}。口语成绩是基于以下六个维度：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，并且有对应的评语&更改建议，在system身份下说明了具体的评分标准。你是一名经验丰富的对外汉语教师，专门负责根据留学生的近期中文口语成绩提供个性化的分析和建议。请根据我近期的中文口语成绩，深入分析我的发音特点，并针对性地提出有效的学习建议。这些建议应考虑到学生的个体差异，既能帮助我们提升口语能力，又具有实际的参考价值和实现可能性。示例：比如有一位留学生近期的中文口语成绩为：{" + exampleScoreList + "}，你的输出示例是：{" + exampleAdvice + "}。";

System.out.println(prompt);

String apiKey = "5176fd66b5ebf072d3a4cb3cc7373e8b.GYxJkFGKPDaaMW3A";

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

JsonObject requestBody = new JsonObject();

requestBody.addProperty("model", "glm-4-flash");

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}));

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();;

Request request = new Request.Builder()

.url(url)

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

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

.build();

String responseBody = null;

String responseAdvice = 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);

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

}

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

System.out.println("Advice: " + responseAdvice);

} else {

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

}

} catch (IOException e) {

throw new RuntimeException(e);

}

adviceActionService.createAdviceAction(userId, responseAdvice);

user.setAdvice(responseAdvice);

if (userMapper.updateById(user) == 0) {

throw new MyException(EnumExceptionType.UPDATE\_FAILED);

}

return new UserInfoResponse(user);

}

@Override

public UserInfoResponse updateAllAdvice() {

return null;

}

}

package fun.redamancyxun.chinese.backend.util;

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

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

import org.springframework.util.CollectionUtils;

import java.util.Collection;

import java.util.Map;

/\*\*

\* @author yannis

\* @version 2020/7/23 0:51

\*/

public class AssertUtil {

public AssertUtil() {

}

public static void isTrue(boolean expValue, EnumExceptionType resultCode, Object obj) {

if (!expValue) {

throw new MyException(resultCode,obj);

}

}

public static void isTrue(boolean expValue, EnumExceptionType resultCode) {

if (!expValue) {

throw new MyException(resultCode);

}

}

public static void isFalse(boolean expValue, EnumExceptionType resultCode, Object obj) {

isTrue(!expValue, resultCode, obj);

}

public static void isFalse(boolean expValue, EnumExceptionType resultCode) {

isTrue(!expValue, resultCode);

}

public static void equals(Object obj1, Object obj2, EnumExceptionType resultCode, Object obj) {

isTrue(obj1 == null ? obj2 == null : obj1.equals(obj2), resultCode, obj);

}

public static void notEquals(Object obj1, Object obj2, EnumExceptionType resultCode, Object obj) {

isTrue(obj1 == null ? obj2 != null : !obj1.equals(obj2), resultCode, obj);

}

public static void contains(Object base, Collection<?> collection, EnumExceptionType resultCode, Object obj) {

notEmpty(collection, resultCode, obj);

isTrue(collection.contains(base), resultCode, obj);

}

public static void contains(Object base, Collection<?> collection, EnumExceptionType resultCode) {

notEmpty(collection, resultCode);

isTrue(collection.contains(base), resultCode);

}

public static void in(Object base, Object[] collection, EnumExceptionType resultCode, Object obj) {

notNull(collection, resultCode, obj);

boolean hasEqual = false;

Object[] var5 = collection;

int var6 = collection.length;

for(int var7 = 0; var7 < var6; ++var7) {

Object obj2 = var5[var7];

if (base == obj2) {

hasEqual = true;

break;

}

}

isTrue(hasEqual, resultCode, obj);

}

public static void notIn(Object base, Object[] collection, EnumExceptionType resultCode, Object obj) {

if (null != collection) {

Object[] var4 = collection;

int var5 = collection.length;

for(int var6 = 0; var6 < var5; ++var6) {

Object obj2 = var4[var6];

isTrue(base != obj2, resultCode, obj);

}

}

}

public static void blank(String str, EnumExceptionType resultCode, Object obj) {

isTrue(isBlank(str), resultCode, obj);

}

public static void notBlank(String str, EnumExceptionType resultCode, Object obj) {

isTrue(!isBlank(str), resultCode, obj);

}

public static void isNull(Object object, EnumExceptionType resultCode, Object obj) {

isTrue(object == null, resultCode, obj);

}

public static void notNull(Object object, EnumExceptionType resultCode, Object obj) {

isTrue(object != null, resultCode, obj);

}

public static void notNull(Object object, EnumExceptionType resultCode) {

isTrue(object != null, resultCode, null);

}

public static void isNull(Object object, EnumExceptionType resultCode) {

isTrue(object == null, resultCode, null);

}

public static void notEmpty(Collection collection, EnumExceptionType resultCode, Object obj) {

isTrue(!CollectionUtils.isEmpty(collection), resultCode, obj);

}

public static void notEmpty(Collection collection, EnumExceptionType resultCode) {

isTrue(!CollectionUtils.isEmpty(collection), resultCode);

}

public static void empty(Collection collection, EnumExceptionType resultCode, Object obj) {

isTrue(CollectionUtils.isEmpty(collection), resultCode, obj);

}

public static void notEmpty(Map map, EnumExceptionType resultCode, Object obj) {

isTrue(!CollectionUtils.isEmpty(map), resultCode, obj);

}

public static void empty(Map map, EnumExceptionType resultCode, Object obj) {

isTrue(CollectionUtils.isEmpty(map), resultCode, obj);

}

private static boolean isBlank(String str) {

int strLen;

if(str != null && (strLen = str.length()) != 0) {

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

if(!Character.isWhitespace(str.charAt(i))) {

return false;

}

}

return true;

} else {

return true;

}

}

}

package fun.redamancyxun.chinese.backend.util;

import com.rabbitmq.client.Connection;

import com.rabbitmq.client.ConnectionFactory;

class ConnectionUtil {

public static Connection getConnection() throws Exception {

ConnectionFactory factory = new ConnectionFactory();

factory.setHost("localhost");

factory.setPort(5672);

factory.setVirtualHost("paperSystem");

factory.setUsername("admin");

factory.setPassword("Lishuai0923");

return factory.newConnection();

}

}

package fun.redamancyxun.chinese.backend.util;

import java.sql.Timestamp;

import java.util.\*;

/\*

\* 由于为了以后使用方便,所有方法的返回类型都设为了 java.util.Date 请在使用时根据自己的需要进行日期格式化处理,如:

\*

\* import java.text.SimpleDateFormat;SimpleDateFormat simpleDateFormat = new

\* SimpleDateFormat("yyyy-MM-dd HH:mm:ss");String todayBegin =

\* simpleDateFormat.format

\* (DateUtils.getDayBegin());System.out.println(todayBegin );//输出结果为2017-10-26

\* 00:00:00

\*/

/\*\*

\* 日期工具类

\*/

public class DatesUtil {

public static Date getDayBegin() {

Calendar cal = new GregorianCalendar();

cal.set(Calendar.HOUR\_OF\_DAY, 0);

cal.set(Calendar.MINUTE, 0);

cal.set(Calendar.SECOND, 0);

cal.set(Calendar.MILLISECOND, 0);

return cal.getTime();

}

public static Date getDayEnd() {

Calendar cal = new GregorianCalendar();

cal.set(Calendar.HOUR\_OF\_DAY, 23);

cal.set(Calendar.MINUTE, 59);

cal.set(Calendar.SECOND, 59);

return cal.getTime();

}

public static Date getBeginDayOfYesterday() {

Calendar cal = new GregorianCalendar();

cal.setTime(getDayBegin());

cal.add(Calendar.DAY\_OF\_MONTH, -1);

return cal.getTime();

}

public static Date getEndDayOfYesterDay() {

Calendar cal = new GregorianCalendar();

cal.setTime(getDayEnd());

cal.add(Calendar.DAY\_OF\_MONTH, -1);

return cal.getTime();

}

public static Date getBeginDayOfTomorrow() {

Calendar cal = new GregorianCalendar();

cal.setTime(getDayBegin());

cal.add(Calendar.DAY\_OF\_MONTH, 1);

return cal.getTime();

}

public static Date getEndDayOfTomorrow() {

Calendar cal = new GregorianCalendar();

cal.setTime(getDayEnd());

cal.add(Calendar.DAY\_OF\_MONTH, 1);

return cal.getTime();

}

@SuppressWarnings("unused")

public static Date getBeginDayOfWeek() {

Date date = new Date();

if (date == null) {

return null;

}

Calendar cal = Calendar.getInstance();

cal.setTime(date);

int dayofweek = cal.get(Calendar.DAY\_OF\_WEEK);

if (dayofweek == 1) {

dayofweek += 7;

}

cal.add(Calendar.DATE, 2 - dayofweek);

return getDayStartTime(cal.getTime());

}

public static Date getEndDayOfWeek() {

Calendar cal = Calendar.getInstance();

cal.setTime(getBeginDayOfWeek());

cal.add(Calendar.DAY\_OF\_WEEK, 6);

Date weekEndSta = cal.getTime();

return getDayEndTime(weekEndSta);

}

@SuppressWarnings("unused")

public static Date getBeginDayOfLastWeek() {

Date date = new Date();

if (date == null) {

return null;

}

Calendar cal = Calendar.getInstance();

cal.setTime(date);

int dayofweek = cal.get(Calendar.DAY\_OF\_WEEK);

if (dayofweek == 1) {

dayofweek += 7;

}

cal.add(Calendar.DATE, 2 - dayofweek - 7);

return getDayStartTime(cal.getTime());

}

public static Date getEndDayOfLastWeek() {

Calendar cal = Calendar.getInstance();

cal.setTime(getBeginDayOfLastWeek());

cal.add(Calendar.DAY\_OF\_WEEK, 6);

Date weekEndSta = cal.getTime();

return getDayEndTime(weekEndSta);

}

public static Date getBeginDayOfMonth() {

Calendar calendar = Calendar.getInstance();

calendar.set(getNowYear(), getNowMonth() - 1, 1);

return getDayStartTime(calendar.getTime());

}

public static Date getEndDayOfMonth() {

Calendar calendar = Calendar.getInstance();

calendar.set(getNowYear(), getNowMonth() - 1, 1);

int day = calendar.getActualMaximum(5);

calendar.set(getNowYear(), getNowMonth() - 1, day);

return getDayEndTime(calendar.getTime());

}

public static Date getBeginDayOfLastMonth() {

Calendar calendar = Calendar.getInstance();

calendar.set(getNowYear(), getNowMonth() - 2, 1);

return getDayStartTime(calendar.getTime());

}

public static Date getEndDayOfLastMonth() {

Calendar calendar = Calendar.getInstance();

calendar.set(getNowYear(), getNowMonth() - 2, 1);

int day = calendar.getActualMaximum(5);

calendar.set(getNowYear(), getNowMonth() - 2, day);

return getDayEndTime(calendar.getTime());

}

public static Date getBeginDayOfYear() {

Calendar cal = Calendar.getInstance();

cal.set(Calendar.YEAR, getNowYear());

cal.set(Calendar.MONTH, Calendar.JANUARY);

cal.set(Calendar.DATE, 1);

return getDayStartTime(cal.getTime());

}

public static Date getEndDayOfYear() {

Calendar cal = Calendar.getInstance();

cal.set(Calendar.YEAR, getNowYear());

cal.set(Calendar.MONTH, Calendar.DECEMBER);

cal.set(Calendar.DATE, 31);

return getDayEndTime(cal.getTime());

}

public static Timestamp getDayStartTime(Date d) {

Calendar calendar = Calendar.getInstance();

if (null != d)

calendar.setTime(d);

calendar.set(calendar.get(Calendar.YEAR), calendar.get(Calendar.MONTH),

calendar.get(Calendar.DAY\_OF\_MONTH), 0, 0, 0);

calendar.set(Calendar.MILLISECOND, 0);

return new Timestamp(calendar.getTimeInMillis());

}

public static Timestamp getDayEndTime(Date d) {

Calendar calendar = Calendar.getInstance();

if (null != d)

calendar.setTime(d);

calendar.set(calendar.get(Calendar.YEAR), calendar.get(Calendar.MONTH),

calendar.get(Calendar.DAY\_OF\_MONTH), 23, 59, 59);

calendar.set(Calendar.MILLISECOND, 999);

return new Timestamp(calendar.getTimeInMillis());

}

public static Integer getNowYear() {

Date date = new Date();

GregorianCalendar gc = (GregorianCalendar) Calendar.getInstance();

gc.setTime(date);

return Integer.valueOf(gc.get(1));

}

public static int getNowMonth() {

Date date = new Date();

GregorianCalendar gc = (GregorianCalendar) Calendar.getInstance();

gc.setTime(date);

return gc.get(2) + 1;

}

public static int getDiffDays(Date beginDate, Date endDate) {

if (beginDate == null || endDate == null) {

throw new IllegalArgumentException("getDiffDays param is null!");

}

long diff = (endDate.getTime() - beginDate.getTime())

/ (1000 \* 60 \* 60 \* 24);

int days = new Long(diff).intValue();

return days;

}

public static long dateDiff(Date beginDate, Date endDate) {

long date1ms = beginDate.getTime();

long date2ms = endDate.getTime();

return date2ms - date1ms;

}

public static Date max(Date beginDate, Date endDate) {

if (beginDate == null) {

return endDate;

}

if (endDate == null) {

return beginDate;

}

if (beginDate.after(endDate)) {

return beginDate;

}

return endDate;

}

public static Date min(Date beginDate, Date endDate) {

if (beginDate == null) {

return endDate;

}

if (endDate == null) {

return beginDate;

}

if (beginDate.after(endDate)) {

return endDate;

}

return beginDate;

}

public static Date getFirstSeasonDate(Date date) {

final int[] SEASON = { 1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4 };

Calendar cal = Calendar.getInstance();

cal.setTime(date);

int sean = SEASON[cal.get(Calendar.MONTH)];

cal.set(Calendar.MONTH, sean \* 3 - 3);

return cal.getTime();

}

public static Date getNextDay(Date date, int i) {

Calendar cal = new GregorianCalendar();

cal.setTime(date);

cal.set(Calendar.DATE, cal.get(Calendar.DATE) + i);

return cal.getTime();

}

public static Date getFrontDay(Date date, int i) {

Calendar cal = new GregorianCalendar();

cal.setTime(date);

cal.set(Calendar.DATE, cal.get(Calendar.DATE) - i);

return cal.getTime();

}

@SuppressWarnings({ "rawtypes", "unchecked" })

public static List getTimeList(int beginYear, int beginMonth, int endYear,

int endMonth, int k) {

List list = new ArrayList();

if (beginYear == endYear) {

for (int j = beginMonth; j <= endMonth; j++) {

list.add(getTimeList(beginYear, j, k));

}

} else {

{

for (int j = beginMonth; j < 12; j++) {

list.add(getTimeList(beginYear, j, k));

}

for (int i = beginYear + 1; i < endYear; i++) {

for (int j = 0; j < 12; j++) {

list.add(getTimeList(i, j, k));

}

}

for (int j = 0; j <= endMonth; j++) {

list.add(getTimeList(endYear, j, k));

}

}

}

return list;

}

@SuppressWarnings({ "unchecked", "rawtypes" })

public static List getTimeList(int beginYear, int beginMonth, int k) {

List list = new ArrayList();

Calendar begincal = new GregorianCalendar(beginYear, beginMonth, 1);

int max = begincal.getActualMaximum(Calendar.DATE);

for (int i = 1; i < max; i = i + k) {

list.add(begincal.getTime());

begincal.add(Calendar.DATE, k);

}

begincal = new GregorianCalendar(beginYear, beginMonth, max);

list.add(begincal.getTime());

return list;

}

}

package fun.redamancyxun.chinese.backend.util;

public class DisTranUtil {

private static final double EARTH\_RADIUS = 6378137;//赤道半径(单位m)

private static final int SPEED = 80000;//假设的货车运送速度(80000km/h)

/\*\*

\* 转化为弧度(rad)

\* \*/

private static double rad(double d)

{

return d \* Math.PI / 180.0;

}

/\*\*

\* 基于余弦定理求两经纬度距离

\* @param lgn1 第一点的精度

\* @param lat1 第一点的纬度

\* @param lgn2 第二点的精度

\* @param lat2 第二点的纬度

\* @return 运送时间，单位：小时

\* \*/

public static Integer tranTime(double lgn1, double lat1, double lgn2, double lat2) {

double radLat1 = rad(lat1);

double radLat2 = rad(lat2);

double radlgn1 = rad(lgn1);

double radlgn2 = rad(lgn2);

if (radLat1 < 0)

radLat1 = Math.PI / 2 + Math.abs(radLat1);// south

if (radLat1 > 0)

radLat1 = Math.PI / 2 - Math.abs(radLat1);// north

if (radlgn1 < 0)

radlgn1 = Math.PI \* 2 - Math.abs(radlgn1);// west

if (radLat2 < 0)

radLat2 = Math.PI / 2 + Math.abs(radLat2);// south

if (radLat2 > 0)

radLat2 = Math.PI / 2 - Math.abs(radLat2);// north

if (radlgn2 < 0)

radlgn2 = Math.PI \* 2 - Math.abs(radlgn2);// west

double x1 = EARTH\_RADIUS \* Math.cos(radlgn1) \* Math.sin(radLat1);

double y1 = EARTH\_RADIUS \* Math.sin(radlgn1) \* Math.sin(radLat1);

double z1 = EARTH\_RADIUS \* Math.cos(radLat1);

double x2 = EARTH\_RADIUS \* Math.cos(radlgn2) \* Math.sin(radLat2);

double y2 = EARTH\_RADIUS \* Math.sin(radlgn2) \* Math.sin(radLat2);

double z2 = EARTH\_RADIUS \* Math.cos(radLat2);

double d = Math.sqrt((x1 - x2) \* (x1 - x2) + (y1 - y2) \* (y1 - y2)+ (z1 - z2) \* (z1 - z2));

double theta = Math.acos((EARTH\_RADIUS \* EARTH\_RADIUS + EARTH\_RADIUS \* EARTH\_RADIUS - d \* d) / (2 \* EARTH\_RADIUS \* EARTH\_RADIUS));

double dist = theta \* EARTH\_RADIUS;

return new Integer((int)(dist/SPEED));

}

}

package fun.redamancyxun.chinese.backend.util;

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

import lombok.\*;

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

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.xssf.usermodel.XSSFWorkbook;

import org.springframework.scheduling.annotation.Scheduled;

import org.springframework.stereotype.Component;

import java.io.FileInputStream;

import java.io.IOException;

import java.util.HashMap;

import java.util.Map;

@Setter

@Getter

public class ExcelReader {

private static ExcelReader excelReaderBeginner = new ExcelReader(0);

private static ExcelReader excelReaderDevelopment = new ExcelReader(1);

private static ExcelReader excelReaderUpperIntermediate = new ExcelReader(2);

private static ExcelReader excelReaderLowerIntermediate = new ExcelReader(3);

private Map<String, String> dataMap = new HashMap<>();

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

private Integer unitCount;

public static ExcelReader getInstance(int num) {

if (num == 0) {

return excelReaderBeginner;

} else if (num == 1) {

return excelReaderDevelopment;

} else if (num == 2) {

return excelReaderUpperIntermediate;

} else {

return excelReaderLowerIntermediate;

}

}

private ExcelReader(int num) {

String excelFilePath = CommonConstants.LINUX\_RESOURCES\_STATIC\_PATH + "lesson.xlsx";

try (FileInputStream fis = new FileInputStream(excelFilePath);

Workbook workbook = new XSSFWorkbook(fis)) {

Sheet sheet = workbook.getSheetAt(num); // 获取工作表

for (Row row : sheet) {

Cell firstCell = row.getCell(0);

Cell thirdCell = row.getCell(2);

if (firstCell != null && thirdCell != null) {

String firstValue = firstCell.getStringCellValue();

String thirdValue = thirdCell.getStringCellValue();

dataMap.put(firstValue, thirdValue);

String category = firstValue.split("-")[0]; // 获取第一列中字符‘-’前的内容

categoryCountMap.put(category, categoryCountMap.getOrDefault(category, 0) + 1);

}

}

unitCount = categoryCountMap.size();

} catch (IOException e) {

e.printStackTrace();

}

System.out.println("Data Map: " + dataMap);

System.out.println("Category Count Map: " + categoryCountMap);

System.out.println("Unit Count: " + unitCount);

}

/\*\*

\* 读取课程信息

\* @throws IOException

\*/

@Scheduled(cron = "0 0 0 \* \* ?") // 每日零点执行

public void updateLesson() throws IOException {

String excelFilePath = CommonConstants.LINUX\_RESOURCES\_STATIC\_PATH + "lesson.xlsx";

try {

FileInputStream fis = new FileInputStream(excelFilePath);

Workbook workbook = new XSSFWorkbook(fis);

for (int i = 0; i < 4; i++) {

Sheet sheet = workbook.getSheetAt(i); // 获取第一个工作表

for (Row row : sheet) {

Cell firstCell = row.getCell(0);

Cell thirdCell = row.getCell(2);

if (firstCell != null && thirdCell != null) {

String firstValue = firstCell.getStringCellValue();

String thirdValue = thirdCell.getStringCellValue();

dataMap.put(firstValue, thirdValue);

String category = firstValue.split("-")[0]; // 获取第一列中字符‘-’前的内容

categoryCountMap.put(category, categoryCountMap.getOrDefault(category, 0) + 1);

}

}

}

unitCount = categoryCountMap.size();

} catch (IOException e) {

e.printStackTrace();

}

System.out.println("Data Map: " + dataMap);

System.out.println("Category Count Map: " + categoryCountMap);

System.out.println("Unit Count: " + unitCount);

}

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

ExcelReader excelReader = ExcelReader.getInstance(3);

excelReader.updateLesson();

System.out.println(excelReader.getDataMap());

System.out.println(excelReader.getCategoryCountMap());

System.out.println(excelReader.getUnitCount());

}

}

package fun.redamancyxun.chinese.backend.util;

import java.io.File;

import java.io.FileOutputStream;

import java.io.InputStream;

import java.io.OutputStream;

public class FileUtil {

public static void inputStreamToFile(InputStream ins, File file) {

try {

OutputStream os = new FileOutputStream(file);

int bytesRead = 0;

byte[] buffer = new byte[8192];

while ((bytesRead = ins.read(buffer, 0, 8192)) != -1) {

os.write(buffer, 0, bytesRead);

}

os.close();

ins.close();

} catch (Exception e) {

e.printStackTrace();

}

}

}

package fun.redamancyxun.chinese.backend.util;

import lombok.extern.slf4j.Slf4j;

import org.apache.http.client.config.RequestConfig;

import org.apache.http.client.entity.UrlEncodedFormEntity;

import org.apache.http.client.methods.CloseableHttpResponse;

import org.apache.http.client.methods.HttpGet;

import org.apache.http.client.methods.HttpPost;

import org.apache.http.entity.ContentType;

import org.apache.http.entity.StringEntity;

import org.apache.http.impl.client.CloseableHttpClient;

import org.apache.http.impl.client.DefaultHttpRequestRetryHandler;

import org.apache.http.impl.client.HttpClients;

import org.apache.http.message.BasicHeader;

import org.apache.http.message.BasicNameValuePair;

import org.apache.http.util.EntityUtils;

import org.slf4j.MDC;

import org.springframework.http.HttpStatus;

import org.springframework.util.CollectionUtils;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import java.util.Map;

/\*\*

\* HTTP/HTTPS 请求封装: GET / POST

\* 默认失败重试3次

\* @author admin

\*/

@Slf4j

public class HttpClientSslUtils {

/\*\*

\* 默认的字符编码格式

\*/

private static final String DEFAULT\_CHAR\_SET = "UTF-8";

/\*\*

\* 默认连接超时时间 (毫秒)

\*/

private static final Integer DEFAULT\_CONNECTION\_TIME\_OUT = 2000;

/\*\*

\* 默认socket超时时间 (毫秒)

\*/

private static final Integer DEFAULT\_SOCKET\_TIME\_OUT = 3000;

/\*\* socketTimeOut上限 \*/

private static final Integer SOCKET\_TIME\_OUT\_UPPER\_LIMIT = 10000;

/\*\* socketTimeOut下限 \*/

private static final Integer SOCKET\_TIME\_OUT\_LOWER\_LIMIT = 1000;

private static CloseableHttpClient getHttpClient() {

RequestConfig requestConfig = RequestConfig.custom().setSocketTimeout(DEFAULT\_SOCKET\_TIME\_OUT)

.setConnectTimeout(DEFAULT\_CONNECTION\_TIME\_OUT).build();

return HttpClients.custom().setDefaultRequestConfig(requestConfig)

.setRetryHandler(new DefaultHttpRequestRetryHandler()).build();

}

private static CloseableHttpClient getHttpClient(Integer socketTimeOut) {

RequestConfig requestConfig =

RequestConfig.custom().setSocketTimeout(socketTimeOut).setConnectTimeout(DEFAULT\_CONNECTION\_TIME\_OUT)

.build();

return HttpClients.custom().setDefaultRequestConfig(requestConfig)

.setRetryHandler(new DefaultHttpRequestRetryHandler()).build();

}

public static String doPost(String url, String requestBody) throws Exception {

return doPost(url, requestBody, ContentType.APPLICATION\_JSON);

}

public static String doPost(String url, String requestBody, Integer socketTimeOut) throws Exception {

return doPost(url, requestBody, ContentType.APPLICATION\_JSON, null, socketTimeOut);

}

public static String doPost(String url, String requestBody, ContentType contentType) throws Exception {

return doPost(url, requestBody, contentType, null);

}

public static String doPost(String url, String requestBody, List<BasicHeader> headers) throws Exception {

return doPost(url, requestBody, ContentType.APPLICATION\_JSON, headers);

}

public static String doPost(String url, String requestBody, ContentType contentType, List<BasicHeader> headers)

throws Exception {

return doPost(url, requestBody, contentType, headers, getHttpClient());

}

public static String doPost(String url, String requestBody, ContentType contentType, List<BasicHeader> headers,

Integer socketTimeOut) throws Exception {

if (socketTimeOut < SOCKET\_TIME\_OUT\_LOWER\_LIMIT || socketTimeOut > SOCKET\_TIME\_OUT\_UPPER\_LIMIT) {

log.error("socketTimeOut非法");

throw new Exception();

}

return doPost(url, requestBody, contentType, headers, getHttpClient(socketTimeOut));

}

/\*\*

\* 通用Post远程服务请求

\* @param url

\* 请求url地址

\* @param requestBody

\* 请求体body

\* @param contentType

\* 内容类型

\* @param headers

\* 请求头

\* @return String 业务自行解析

\* @throws Exception

\*/

public static String doPost(String url, String requestBody, ContentType contentType, List<BasicHeader> headers,

CloseableHttpClient client) throws Exception {

CloseableHttpResponse response = null;

long startTime = System.currentTimeMillis();

try {

HttpPost post = new HttpPost(url);

if (!CollectionUtils.isEmpty(headers)) {

for (BasicHeader header : headers) {

post.setHeader(header);

}

}

StringEntity entity =

new StringEntity(requestBody, ContentType.create(contentType.getMimeType(), DEFAULT\_CHAR\_SET));

post.setEntity(entity);

response = client.execute(post);

if (response.getStatusLine().getStatusCode() != HttpStatus.OK.value()) {

log.error("业务请求返回失败:{}", EntityUtils.toString(response.getEntity()));

throw new Exception();

}

return EntityUtils.toString(response.getEntity());

} finally {

releaseResourceAndLog(url, requestBody, response, startTime);

}

}

/\*\*

\* 暂时用于智慧园区业务联调方式

\* @param url 业务请求url

\* @param param 业务参数

\* @return

\* @throws Exception

\*/

public static String doPostWithUrlEncoded(String url,

Map<String, String> param) throws Exception {

CloseableHttpClient httpClient = getHttpClient();

CloseableHttpResponse response = null;

long startTime = System.currentTimeMillis();

try {

HttpPost httpPost = new HttpPost(url);

if (param != null) {

List<org.apache.http.NameValuePair> paramList = new ArrayList<>();

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

paramList.add(new BasicNameValuePair(key, param.get(key)));

}

UrlEncodedFormEntity entity = new UrlEncodedFormEntity(paramList, DEFAULT\_CHAR\_SET);

httpPost.setEntity(entity);

}

response = httpClient.execute(httpPost);

if (response.getStatusLine().getStatusCode() != HttpStatus.OK.value()) {

log.error("业务请求返回失败:{}" , EntityUtils.toString(response.getEntity()));

throw new Exception();

}

return EntityUtils.toString(response.getEntity(), DEFAULT\_CHAR\_SET);

} finally {

releaseResourceAndLog(url, param == null ? null : param.toString(), response, startTime);

}

}

private static void releaseResourceAndLog(String url, String request, CloseableHttpResponse response, long startTime) {

if (null != response) {

try {

response.close();

recordInterfaceLog(startTime, url, request);

} catch (IOException e) {

log.error(e.getMessage());

}

}

}

public static String doGet(String url) throws Exception {

return doGet(url, ContentType.DEFAULT\_TEXT);

}

public static String doGet(String url, ContentType contentType) throws Exception {

return doGet(url, contentType, null);

}

public static String doGet(String url, List<BasicHeader> headers) throws Exception {

return doGet(url, ContentType.DEFAULT\_TEXT, headers);

}

/\*\*

\* 通用Get远程服务请求

\* @param url

\* 请求参数

\* @param contentType

\* 请求参数类型

\* @param headers

\* 请求头可以填充

\* @return String 业务自行解析数据

\* @throws Exception

\*/

public static String doGet(String url, ContentType contentType, List<BasicHeader> headers) throws Exception {

CloseableHttpResponse response = null;

long startTime = System.currentTimeMillis();

try {

CloseableHttpClient client = getHttpClient();

HttpGet httpGet = new HttpGet(url);

if (!CollectionUtils.isEmpty(headers)) {

for (BasicHeader header : headers) {

httpGet.setHeader(header);

}

}

if(contentType != null){

httpGet.setHeader("Content-Type", contentType.getMimeType());

}

response = client.execute(httpGet);

if (response.getStatusLine().getStatusCode() != HttpStatus.OK.value()) {

log.error("业务请求返回失败:{}", EntityUtils.toString(response.getEntity()));

throw new Exception();

}

return EntityUtils.toString(response.getEntity());

} finally {

releaseResourceAndLog(url, null, response, startTime);

}

}

private static void recordInterfaceLog(long startTime, String url, String request) {

long endTime = System.currentTimeMillis();

long timeCost = endTime - startTime;

MDC.put("totalTime", String.valueOf(timeCost));

MDC.put("url", url);

MDC.put("logType", "third-platform-service");

log.info("HttpClientSslUtils 远程请求:{} 参数:{} 耗时:{}ms", url, request, timeCost);

}

}

package fun.redamancyxun.chinese.backend.util;

import com.alibaba.fastjson.JSONObject;

import lombok.extern.slf4j.Slf4j;

import org.apache.http.HttpEntity;

import org.apache.http.HttpResponse;

import org.apache.http.NameValuePair;

import org.apache.http.client.ClientProtocolException;

import org.apache.http.client.HttpClient;

import org.apache.http.client.config.RequestConfig;

import org.apache.http.client.entity.UrlEncodedFormEntity;

import org.apache.http.client.methods.CloseableHttpResponse;

import org.apache.http.client.methods.HttpGet;

import org.apache.http.client.methods.HttpPost;

import org.apache.http.entity.StringEntity;

import org.apache.http.entity.mime.MultipartEntityBuilder;

import org.apache.http.entity.mime.content.FileBody;

import org.apache.http.impl.client.CloseableHttpClient;

import org.apache.http.impl.client.HttpClients;

import org.apache.http.message.BasicNameValuePair;

import org.apache.http.util.EntityUtils;

import java.io.File;

import java.io.IOException;

import java.io.UnsupportedEncodingException;

import java.util.\*;

import java.util.Map.Entry;

import java.util.stream.Collectors;

@Slf4j

public class HttpUtil {

public static String get(String url, Map<String, String> param) {

CloseableHttpClient httpClient = null;

CloseableHttpResponse response = null;

String result = "";

try {

httpClient = HttpClients.createDefault();

url += "?";

url += String.join("&", param.entrySet().stream().map(e -> e.getKey() + "=" + e.getValue()).collect(Collectors.toList()));

log.info("request: " + url);

HttpGet httpGet = new HttpGet(url);

RequestConfig requestConfig = RequestConfig.custom().setConnectTimeout(35000)// 连接主机服务超时时间

.setConnectionRequestTimeout(35000)// 请求超时时间

.setSocketTimeout(60000)// 数据读取超时时间

.build();

httpGet.setConfig(requestConfig);

response = httpClient.execute(httpGet);

HttpEntity entity = response.getEntity();

result = EntityUtils.toString(entity);

} catch (ClientProtocolException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

} finally {

if (null != response) {

try {

response.close();

} catch (IOException e) {

e.printStackTrace();

}

}

if (null != httpClient) {

try {

httpClient.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

return result;

}

public static String post(String url, Map<String, Object> paramMap) {

CloseableHttpClient httpClient = null;

CloseableHttpResponse httpResponse = null;

String result = "";

httpClient = HttpClients.createDefault();

HttpPost httpPost = new HttpPost(url);

RequestConfig requestConfig = RequestConfig.custom().setConnectTimeout(35000)// 设置连接主机服务超时时间

.setConnectionRequestTimeout(35000)// 设置连接请求超时时间

.setSocketTimeout(60000)// 设置读取数据连接超时时间

.build();

httpPost.setConfig(requestConfig);

httpPost.addHeader("Content-Type", "application/x-www-form-urlencoded");

if (null != paramMap && paramMap.size() > 0) {

List<NameValuePair> nvps = new ArrayList<NameValuePair>();

Set<Entry<String, Object>> entrySet = paramMap.entrySet();

Iterator<Entry<String, Object>> iterator = entrySet.iterator();

while (iterator.hasNext()) {

Entry<String, Object> mapEntry = iterator.next();

nvps.add(new BasicNameValuePair(mapEntry.getKey(), mapEntry.getValue().toString()));

}

try {

httpPost.setEntity(new UrlEncodedFormEntity(nvps, "UTF-8"));

} catch (UnsupportedEncodingException e) {

e.printStackTrace();

}

}

try {

httpResponse = httpClient.execute(httpPost);

HttpEntity entity = httpResponse.getEntity();

result = EntityUtils.toString(entity);

} catch (ClientProtocolException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

} finally {

if (null != httpResponse) {

try {

httpResponse.close();

} catch (IOException e) {

e.printStackTrace();

}

}

if (null != httpClient) {

try {

httpClient.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

return result;

}

public static String postJson(String url, Map<String, Object> paramMap, Map<String, Object> jsonMap) {

return postJson(url,paramMap,jsonMap,null);

}

public static String postJson(String url, Map<String, Object> paramMap, Map<String, Object> jsonMap, HttpClient httpClient) {

if(paramMap != null) {

url += "?";

url += String.join("&", paramMap.entrySet().stream().map(e -> e.getKey() + "=" + e.getValue()).collect(Collectors.toList()));

}

if(httpClient == null) {

httpClient = HttpClients.createDefault();

}

HttpPost request = new HttpPost(url);

request.setHeader("Content-Type", "application/json;charset=UTF-8");

JSONObject postData = new JSONObject();

postData.putAll(jsonMap);

String jsonString = postData.toString();

request.setEntity(new StringEntity(jsonString, "utf-8"));

try {

HttpResponse response = httpClient.execute(request);

HttpEntity entity = response.getEntity();

return EntityUtils.toString(entity, "utf-8");

} catch (IOException e) {

e.printStackTrace();

}

return null;

}

public static String postFormData(String url, Map<String, Object> paramMap, Map<String, File> fileMap) {

CloseableHttpClient httpclient = HttpClients.createDefault();

try {

url += "?";

url += String.join("&", paramMap.entrySet().stream().map(e -> e.getKey() + "=" + e.getValue()).collect(Collectors.toList()));

HttpPost httppost = new HttpPost(url);

MultipartEntityBuilder builder = MultipartEntityBuilder.create();

fileMap.forEach((key, value) -> builder.addPart(key, new FileBody(value)));

HttpEntity httpEntity = builder.build();

httppost.setEntity(httpEntity);

String result = null;

CloseableHttpResponse response = httpclient.execute(httppost);

try {

HttpEntity resEntity = response.getEntity();

if (resEntity != null) {

result = EntityUtils.toString(resEntity);

}

EntityUtils.consume(resEntity);

return result;

} finally {

response.close();

}

} catch (Exception e) {

} finally {

try {

httpclient.close();

} catch (IOException e) {

e.printStackTrace();

}

}

return null;

}

public static void main(String[] args) {

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

map.put("c", "f");

map.put("encode", "text");

System.out.println(get("https://v1.hitokoto.cn/",map));

}

}

package fun.redamancyxun.chinese.backend.util;

import javax.imageio.ImageIO;

import java.awt.\*;

import java.awt.color.ColorSpace;

import java.awt.geom.AffineTransform;

import java.awt.image.\*;

import java.io.File;

import java.io.IOException;

/\*\*

\* 图片处理工具类：<br>

\* 功能：缩放图像、切割图像、图像类型转换、彩色转黑白、文字水印、图片水印等

\*

\* @author Administrator

\*/

public class ImageUtil {

/\*\*

\* 几种常见的图片格式

\*/

public static String IMAGE\_TYPE\_GIF = "gif";// 图形交换格式

public static String IMAGE\_TYPE\_JPG = "jpg";// 联合照片专家组

public static String IMAGE\_TYPE\_JPEG = "jpeg";// 联合照片专家组

public static String IMAGE\_TYPE\_BMP = "bmp";// 英文Bitmap（位图）的简写，它是Windows操作系统中的标准图像文件格式

public static String IMAGE\_TYPE\_PNG = "png";// 可移植网络图形

public static String IMAGE\_TYPE\_PSD = "psd";// Photoshop的专用格式Photoshop

public static void main(String[] args) {

String rootFilePath = "d:/1.png";

File image = new File(rootFilePath);

if(image.length() >= 1024){

while(image.length() >= 1024){

scale(rootFilePath, rootFilePath,2,false);

}

}

}

/\*\*

\* 缩放图像（按比例缩放）

\*

\* @param srcImageFile 源图像文件地址

\* @param result 缩放后的图像地址

\* @param scale 缩放比例

\* @param flag 缩放选择:true 放大; false 缩小;

\*/

public static void scale(String srcImageFile, String result,

int scale, boolean flag) {

try {

BufferedImage src = ImageIO.read(new File(srcImageFile)); // 读入文件

int width = src.getWidth(); // 得到源图宽

int height = src.getHeight(); // 得到源图长

if (flag) {// 放大

width = width \* scale;

height = height \* scale;

} else {// 缩小

width = width / scale;

height = height / scale;

}

Image image = src.getScaledInstance(width, height,

Image.SCALE\_DEFAULT);

BufferedImage tag = new BufferedImage(width, height,

BufferedImage.TYPE\_INT\_RGB);

Graphics g = tag.getGraphics();

g.drawImage(image, 0, 0, null); // 绘制缩小后的图

g.dispose();

ImageIO.write(tag, "JPEG", new File(result));// 输出到文件流

} catch (IOException e) {

e.printStackTrace();

}

}

/\*\*

\* 缩放图像（按高度和宽度缩放）

\*

\* @param srcImageFile 源图像文件地址

\* @param result 缩放后的图像地址

\* @param height 缩放后的高度

\* @param width 缩放后的宽度

\* @param bb 比例不对时是否需要补白：true为补白; false为不补白;

\*/

public final static void scale2(String srcImageFile, String result, int height, int width, boolean bb) {

try {

double ratio = 0.0; // 缩放比例

File f = new File(srcImageFile);

BufferedImage bi = ImageIO.read(f);

Image itemp = bi.getScaledInstance(width, height, bi.SCALE\_SMOOTH);

if ((bi.getHeight() > height) || (bi.getWidth() > width)) {

if (bi.getHeight() > bi.getWidth()) {

ratio = (new Integer(height)).doubleValue()

/ bi.getHeight();

} else {

ratio = (new Integer(width)).doubleValue() / bi.getWidth();

}

AffineTransformOp op = new AffineTransformOp(AffineTransform

.getScaleInstance(ratio, ratio), null);

itemp = op.filter(bi, null);

}

if (bb) {//补白

BufferedImage image = new BufferedImage(width, height,

BufferedImage.TYPE\_INT\_RGB);

Graphics2D g = image.createGraphics();

g.setColor(Color.white);

g.fillRect(0, 0, width, height);

if (width == itemp.getWidth(null))

g.drawImage(itemp, 0, (height - itemp.getHeight(null)) / 2,

itemp.getWidth(null), itemp.getHeight(null),

Color.white, null);

else

g.drawImage(itemp, (width - itemp.getWidth(null)) / 2, 0,

itemp.getWidth(null), itemp.getHeight(null),

Color.white, null);

g.dispose();

itemp = image;

}

ImageIO.write((BufferedImage) itemp, "JPEG", new File(result));

} catch (IOException e) {

e.printStackTrace();

}

}

/\*\*

\* 图像切割(按指定起点坐标和宽高切割)

\*

\* @param srcImageFile 源图像地址

\* @param result 切片后的图像地址

\* @param x 目标切片起点坐标X

\* @param y 目标切片起点坐标Y

\* @param width 目标切片宽度

\* @param height 目标切片高度

\*/

public final static void cut(String srcImageFile, String result,

int x, int y, int width, int height) {

try {

BufferedImage bi = ImageIO.read(new File(srcImageFile));

int srcWidth = bi.getHeight(); // 源图宽度

int srcHeight = bi.getWidth(); // 源图高度

if (srcWidth > 0 && srcHeight > 0) {

Image image = bi.getScaledInstance(srcWidth, srcHeight,

Image.SCALE\_DEFAULT);

ImageFilter cropFilter = new CropImageFilter(x, y, width, height);

Image img = Toolkit.getDefaultToolkit().createImage(

new FilteredImageSource(image.getSource(),

cropFilter));

BufferedImage tag = new BufferedImage(width, height, BufferedImage.TYPE\_INT\_RGB);

Graphics g = tag.getGraphics();

g.drawImage(img, 0, 0, width, height, null); // 绘制切割后的图

g.dispose();

ImageIO.write(tag, "JPEG", new File(result));

}

} catch (Exception e) {

e.printStackTrace();

}

}

/\*\*

\* 图像切割（指定切片的行数和列数）

\*

\* @param srcImageFile 源图像地址

\* @param descDir 切片目标文件夹

\* @param rows 目标切片行数。默认2，必须是范围 [1, 20] 之内

\* @param cols 目标切片列数。默认2，必须是范围 [1, 20] 之内

\*/

public final static void cut2(String srcImageFile, String descDir,

int rows, int cols) {

try {

if (rows <= 0 || rows > 20) rows = 2; // 切片行数

if (cols <= 0 || cols > 20) cols = 2; // 切片列数

BufferedImage bi = ImageIO.read(new File(srcImageFile));

int srcWidth = bi.getHeight(); // 源图宽度

int srcHeight = bi.getWidth(); // 源图高度

if (srcWidth > 0 && srcHeight > 0) {

Image img;

ImageFilter cropFilter;

Image image = bi.getScaledInstance(srcWidth, srcHeight, Image.SCALE\_DEFAULT);

int destWidth = srcWidth; // 每张切片的宽度

int destHeight = srcHeight; // 每张切片的高度

if (srcWidth % cols == 0) {

destWidth = srcWidth / cols;

} else {

destWidth = (int) Math.floor(srcWidth / cols) + 1;

}

if (srcHeight % rows == 0) {

destHeight = srcHeight / rows;

} else {

destHeight = (int) Math.floor(srcWidth / rows) + 1;

}

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

for (int j = 0; j < cols; j++) {

cropFilter = new CropImageFilter(j \* destWidth, i \* destHeight,

destWidth, destHeight);

img = Toolkit.getDefaultToolkit().createImage(

new FilteredImageSource(image.getSource(),

cropFilter));

BufferedImage tag = new BufferedImage(destWidth,

destHeight, BufferedImage.TYPE\_INT\_RGB);

Graphics g = tag.getGraphics();

g.drawImage(img, 0, 0, null); // 绘制缩小后的图

g.dispose();

ImageIO.write(tag, "JPEG", new File(descDir

+ "\_r" + i + "\_c" + j + ".jpg"));

}

}

}

} catch (Exception e) {

e.printStackTrace();

}

}

/\*\*

\* 图像切割（指定切片的宽度和高度）

\*

\* @param srcImageFile 源图像地址

\* @param descDir 切片目标文件夹

\* @param destWidth 目标切片宽度。默认200

\* @param destHeight 目标切片高度。默认150

\*/

public final static void cut3(String srcImageFile, String descDir,

int destWidth, int destHeight) {

try {

if (destWidth <= 0) destWidth = 200; // 切片宽度

if (destHeight <= 0) destHeight = 150; // 切片高度

BufferedImage bi = ImageIO.read(new File(srcImageFile));

int srcWidth = bi.getHeight(); // 源图宽度

int srcHeight = bi.getWidth(); // 源图高度

if (srcWidth > destWidth && srcHeight > destHeight) {

Image img;

ImageFilter cropFilter;

Image image = bi.getScaledInstance(srcWidth, srcHeight, Image.SCALE\_DEFAULT);

int cols = 0; // 切片横向数量

int rows = 0; // 切片纵向数量

if (srcWidth % destWidth == 0) {

cols = srcWidth / destWidth;

} else {

cols = (int) Math.floor(srcWidth / destWidth) + 1;

}

if (srcHeight % destHeight == 0) {

rows = srcHeight / destHeight;

} else {

rows = (int) Math.floor(srcHeight / destHeight) + 1;

}

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

for (int j = 0; j < cols; j++) {

cropFilter = new CropImageFilter(j \* destWidth, i \* destHeight,

destWidth, destHeight);

img = Toolkit.getDefaultToolkit().createImage(

new FilteredImageSource(image.getSource(),

cropFilter));

BufferedImage tag = new BufferedImage(destWidth,

destHeight, BufferedImage.TYPE\_INT\_RGB);

Graphics g = tag.getGraphics();

g.drawImage(img, 0, 0, null); // 绘制缩小后的图

g.dispose();

ImageIO.write(tag, "JPEG", new File(descDir

+ "\_r" + i + "\_c" + j + ".jpg"));

}

}

}

} catch (Exception e) {

e.printStackTrace();

}

}

/\*\*

\* 图像类型转换：GIF->JPG、GIF->PNG、PNG->JPG、PNG->GIF(X)、BMP->PNG

\*

\* @param srcImageFile 源图像地址

\* @param formatName 包含格式非正式名称的 String：如JPG、JPEG、GIF等

\* @param destImageFile 目标图像地址

\*/

public final static void convert(String srcImageFile, String formatName, String destImageFile) {

try {

File f = new File(srcImageFile);

f.canRead();

f.canWrite();

BufferedImage src = ImageIO.read(f);

ImageIO.write(src, formatName, new File(destImageFile));

} catch (Exception e) {

e.printStackTrace();

}

}

/\*\*

\* 彩色转为黑白

\*

\* @param srcImageFile 源图像地址

\* @param destImageFile 目标图像地址

\*/

public final static void gray(String srcImageFile, String destImageFile) {

try {

BufferedImage src = ImageIO.read(new File(srcImageFile));

ColorSpace cs = ColorSpace.getInstance(ColorSpace.CS\_GRAY);

ColorConvertOp op = new ColorConvertOp(cs, null);

src = op.filter(src, null);

ImageIO.write(src, "JPEG", new File(destImageFile));

} catch (IOException e) {

e.printStackTrace();

}

}

/\*\*

\* 给图片添加文字水印

\*

\* @param pressText 水印文字

\* @param srcImageFile 源图像地址

\* @param destImageFile 目标图像地址

\* @param fontName 水印的字体名称

\* @param fontStyle 水印的字体样式

\* @param color 水印的字体颜色

\* @param fontSize 水印的字体大小

\* @param x 修正值

\* @param y 修正值

\* @param alpha 透明度：alpha 必须是范围 [0.0, 1.0] 之内（包含边界值）的一个浮点数字

\*/

public final static void pressText(String pressText,

String srcImageFile, String destImageFile, String fontName,

int fontStyle, Color color, int fontSize, int x,

int y, float alpha) {

try {

File img = new File(srcImageFile);

Image src = ImageIO.read(img);

int width = src.getWidth(null);

int height = src.getHeight(null);

BufferedImage image = new BufferedImage(width, height,

BufferedImage.TYPE\_INT\_RGB);

Graphics2D g = image.createGraphics();

g.drawImage(src, 0, 0, width, height, null);

g.setColor(color);

g.setFont(new Font(fontName, fontStyle, fontSize));

g.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC\_ATOP,

alpha));

g.drawString(pressText, (width - (getLength(pressText) \* fontSize))

/ 2 + x, (height - fontSize) / 2 + y);

g.dispose();

ImageIO.write((BufferedImage) image, "JPEG", new File(destImageFile));// 输出到文件流

} catch (Exception e) {

e.printStackTrace();

}

}

/\*\*

\* 给图片添加文字水印

\*

\* @param pressText 水印文字

\* @param srcImageFile 源图像地址

\* @param destImageFile 目标图像地址

\* @param fontName 字体名称

\* @param fontStyle 字体样式

\* @param color 字体颜色

\* @param fontSize 字体大小

\* @param x 修正值

\* @param y 修正值

\* @param alpha 透明度：alpha 必须是范围 [0.0, 1.0] 之内（包含边界值）的一个浮点数字

\*/

public final static void pressText2(String pressText, String srcImageFile, String destImageFile,

String fontName, int fontStyle, Color color, int fontSize, int x,

int y, float alpha) {

try {

File img = new File(srcImageFile);

Image src = ImageIO.read(img);

int width = src.getWidth(null);

int height = src.getHeight(null);

BufferedImage image = new BufferedImage(width, height,

BufferedImage.TYPE\_INT\_RGB);

Graphics2D g = image.createGraphics();

g.drawImage(src, 0, 0, width, height, null);

g.setColor(color);

g.setFont(new Font(fontName, fontStyle, fontSize));

g.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC\_ATOP,

alpha));

g.drawString(pressText, (width - (getLength(pressText) \* fontSize))

/ 2 + x, (height - fontSize) / 2 + y);

g.dispose();

ImageIO.write((BufferedImage) image, "JPEG", new File(destImageFile));

} catch (Exception e) {

e.printStackTrace();

}

}

/\*\*

\* 给图片添加图片水印

\*

\* @param pressImg 水印图片

\* @param srcImageFile 源图像地址

\* @param destImageFile 目标图像地址

\* @param x 修正值。 默认在中间

\* @param y 修正值。 默认在中间

\* @param alpha 透明度：alpha 必须是范围 [0.0, 1.0] 之内（包含边界值）的一个浮点数字

\*/

public final static void pressImage(String pressImg, String srcImageFile, String destImageFile,

int x, int y, float alpha) {

try {

File img = new File(srcImageFile);

Image src = ImageIO.read(img);

int wideth = src.getWidth(null);

int height = src.getHeight(null);

BufferedImage image = new BufferedImage(wideth, height,

BufferedImage.TYPE\_INT\_RGB);

Graphics2D g = image.createGraphics();

g.drawImage(src, 0, 0, wideth, height, null);

Image src\_biao = ImageIO.read(new File(pressImg));

int wideth\_biao = src\_biao.getWidth(null);

int height\_biao = src\_biao.getHeight(null);

g.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC\_ATOP,

alpha));

g.drawImage(src\_biao, (wideth - wideth\_biao) / 2,

(height - height\_biao) / 2, wideth\_biao, height\_biao, null);

g.dispose();

ImageIO.write((BufferedImage) image, "JPEG", new File(destImageFile));

} catch (Exception e) {

e.printStackTrace();

}

}

/\*\*

\* 计算text的长度（一个中文算两个字符）

\*

\* @param text

\* @return

\*/

public final static int getLength(String text) {

int length = 0;

for (int i = 0; i < text.length(); i++) {

if (new String(text.charAt(i) + "").getBytes().length > 1) {

length += 2;

} else {

length += 1;

}

}

return length / 2;

}

}

package fun.redamancyxun.chinese.backend.util;

import com.alibaba.fastjson.JSON;

import com.alibaba.fastjson.JSONObject;

import java.util.List;

import java.util.Map;

/\*\*

\* @author yannis

\* @version 2020/8/1 13:17

\*/

public class JsonUtil {

/\*\*

\* 将一个对象装换为Json字符串

\*/

public static String toJSONString(Object object) {

return JSONObject.toJSONString(object);

}

/\*\*

\* 将Json字符串转换为Object类型的

\*/

public static Object toObject(String str) {

return JSON.parse(str);

}

/\*\*

\* 将Json字符串转换为实例

\*/

public static <T> T toObject(String str, Class<T> t) {

return JSON.parseObject(str, t);

}

/\*\*

\* 将Json转换为指定类型的List

\*/

public static <T> List<T> toList(String str, Class<T> t) {

return JSON.parseArray(str, t);

}

/\*\*

\* 将Json转换为Map

\*/

@SuppressWarnings("unchecked")

public static <T> Map<String, T> jsonToMap(String str) {

return (Map<String, T>) JSONObject.parseObject(str);

}

/\*\*

\* 将Object转换为Map

\*/

@SuppressWarnings("unchecked")

public static <T> Map<String, T> objectToMap(Object object) {

return (Map<String, T>) JSONObject.parseObject(JSONObject.toJSONString(object));

}

}

package fun.redamancyxun.chinese.backend.util;

import org.springframework.mail.javamail.JavaMailSender;

import org.springframework.mail.javamail.MimeMessageHelper;

import org.springframework.stereotype.Component;

import javax.mail.internet.MimeMessage;

import java.io.File;

import java.sql.Timestamp;

import java.util.List;

import static fun.redamancyxun.chinese.backend.common.CommonConstants.IMAGE\_PATH;

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

@Component

public class MessageUtil {

public String signUp(String email, String code) {

String htmlContent = "<!DOCTYPE html>\n" +

"<html>\n" +

" <head>\n" +

" <title>验证码邮件</title>\n" +

" </head>\n" +

" <body style=\"font-family: Arial; color: #333333\">\n" +

" <div\n" +

" style=\"\n" +

" max-width: 600px;\n" +

" margin: 0 auto;\n" +

" background-color: #ffffff;\n" +

" padding: 20px;\n" +

" \"\n" +

" >\n" +

" <h2 style=\"color: #b70031\">您好，<span style=\"font-size: 16px;\">%s</span></h2>\n" +

" <p>\n" +

" 您于 %s 尝试验证此邮箱地址，验证码为：<span style=\"font-size: 24px; color: #b70031; font-weight: 700\">%s</span>\n" +

" </p>\n" +

" <p>请输入此验证码完成操作，验证码15分钟内有效。</p>\n" +

" <p style=\"font-size: 14px; color: #888888; margin-top: 20px\">\n" +

" 如果您没有进行相关操作，请忽略此邮件。<br />\n" +

" 感谢您的使用!\n" +

" </p>\n" +

" </div>\n" +

" </body>\n" +

"</html>";

return String.format(htmlContent, email, getNowTime(), code);

}

/\*\*

\* @param type 0:系统 1:给帖子点赞 2:给帖子回复 3:给评论点赞 4:给评论回复 5:给回复点赞 6:给回复回复

\* @return 返回消息内容

\*/

public static String getNoticeMessage(String sender, Integer type, String content) {

String result;

switch (type) {

case 0:

result = "有新的系统消息";

break;

case 1:

case 3:

case 5:

result = "收到点赞";

break;

case 2:

case 4:

case 6:

result = content;

break;

default:

result = "有新消息";

break;

}

return result;

}

private static String getNowTime() {

Timestamp timestamp = new Timestamp(System.currentTimeMillis());

return timestamp

.toString()

.substring(0, timestamp.toString().indexOf("."));

}

public void sendMail(String sender, String email, String title, String content, List<String> attachments, JavaMailSender jms) throws Exception {

MimeMessage mainMessage = jms.createMimeMessage();

MimeMessageHelper helper = new MimeMessageHelper(mainMessage, true);

helper.setFrom(sender);

helper.setTo(email);

helper.setSubject(title);

helper.setText(content, true);

if (attachments != null && !attachments.isEmpty()) {

for (String attachment : attachments) {

attachment = attachment.replace(IMAGE\_PATH, USER\_FILE\_PATH);

String fileName = attachment.substring(attachment.lastIndexOf(File.separator) + 1);

helper.addAttachment(fileName, new File(attachment));

}

}

jms.send(mainMessage);

}

}

package fun.redamancyxun.chinese.backend.util;

import com.aliyuncs.utils.StringUtils;

import com.fasterxml.jackson.annotation.JsonInclude;

import com.fasterxml.jackson.core.JsonProcessingException;

import com.fasterxml.jackson.databind.DeserializationFeature;

import com.fasterxml.jackson.databind.JsonNode;

import com.fasterxml.jackson.databind.ObjectMapper;

import com.fasterxml.jackson.databind.SerializationFeature;

import lombok.extern.slf4j.Slf4j;

import java.text.SimpleDateFormat;

@Slf4j

public class NewJsonUtil {

/\*\*

\* 定义映射对象

\*/

public static ObjectMapper objectMapper = new ObjectMapper();

/\*\*

\* 日期格式化

\*/

private static final String DATE\_FORMAT = "yyyy-MM-dd HH:mm:ss";

static {

objectMapper.setSerializationInclusion(JsonInclude.Include.NON\_NULL);

objectMapper.configure(SerializationFeature.WRITE\_DATES\_AS\_TIMESTAMPS, false);

objectMapper.configure(SerializationFeature.FAIL\_ON\_EMPTY\_BEANS, false);

objectMapper.setDateFormat(new SimpleDateFormat(DATE\_FORMAT));

objectMapper.configure(DeserializationFeature.FAIL\_ON\_UNKNOWN\_PROPERTIES, false);

}

/\*\*

\* string转JsonNode

\*

\* @param jsonString

\* @return com.fasterxml.jackson.databind.JsonNode

\*/

public static JsonNode stringToJsonNode(String jsonString) throws JsonProcessingException {

return objectMapper.readTree(jsonString);

}

/\*\*

\* 对象转json字符串

\*

\* @param obj

\* @param <T>

\*/

public static <T> String objToString(T obj) {

if (obj == null) {

return null;

}

try {

return obj instanceof String ? (String) obj : objectMapper.writeValueAsString(obj);

} catch (JsonProcessingException e) {

log.warn("Parse Object to String error : {}", e.getMessage());

return null;

}

}

/\*\*

\* 对象转格式化的字符串字符串

\*

\* @param obj

\* @param <T>

\* @return

\*/

public static <T> String objToPrettyString(T obj) {

if (obj == null) {

return null;

}

try {

return obj instanceof String ? (String) obj : objectMapper.writerWithDefaultPrettyPrinter().writeValueAsString(obj);

} catch (JsonProcessingException e) {

log.warn("Parse Object to String error : {}", e.getMessage());

return null;

}

}

/\*\*

\* json字符串转对象

\*

\* @param jsonString

\* @param cls

\* @param <T>

\*/

public static <T> T stringToObj(String jsonString, Class<T> cls) {

if (StringUtils.isEmpty(jsonString) || cls == null) {

return null;

}

try {

return cls.equals(String.class) ? (T) jsonString : objectMapper.readValue(jsonString, cls);

} catch (JsonProcessingException e) {

log.warn("Parse String to Object error : {}", e.getMessage());

return null;

}

}

}

package fun.redamancyxun.chinese.backend.util;

import cn.hutool.core.collection.CollUtil;

import com.github.pagehelper.Page;

import java.util.ArrayList;

import java.util.List;

public class PageUtils {

/\*\*

\* @param pageNum 页码

\* @param pageSize 每页数量

\* @param list 数据

\* 切割分页数据，手动制造分页

\*/

public static <T> List<T> splitList(int pageNum, int pageSize, List<T> list) {

if (list.size() < (pageNum \* pageSize)) {

if(list.size() <= ((pageNum - 1) \* pageSize)){

return new ArrayList<>();

}

if (list.size() % pageSize == 0) {

return list.subList((list.size() / pageSize - 1) \* pageSize, list.size());

}

return list.subList(list.size() / pageSize \* pageSize, list.size());

}

return list.subList((pageNum - 1) \* pageSize, pageNum \* pageSize);

}

/\*\*

\* @param page 分页对象

\* @param pageNum 页码

\* @param pageSize 每页数量

\* @param list 数据

\* 设置分页信息

\*/

public static <T> List<T> setPage(Page<T> page, int pageNum, int pageSize, List<T> list) {

if (CollUtil.isEmpty(list)) {

page.setTotal(0);

return list;

}

List<T> result = splitList(pageNum,pageSize,list);

int total = list.size();

page.setTotal(total);

return result;

}

}

package fun.redamancyxun.chinese.backend.util;

import cn.hutool.crypto.SecureUtil;

public class PasswordUtil {

public static String convert(String origin){

return SecureUtil

.md5(origin)

.substring(3,13);

}

public static void main(String[] args) {

String password = "123456";

System.out.println(SecureUtil.md5(password));

System.out.println(convert(password));

}

}

package fun.redamancyxun.chinese.backend.util;

public class RandomVerifyCodeUtil {

public static int getRandNum(int min, int max) {

int randNum = min + (int)(Math.random() \* ((max - min) + 1));

return randNum;

}

public static String getRandomVerifyCode(){

return getRandNum(100000,999999)+"";

}

}

package fun.redamancyxun.chinese.backend.util;

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

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

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

import org.springframework.stereotype.Component;

import org.springframework.util.CollectionUtils;

import java.util.Collection;

import java.util.List;

import java.util.Map;

import java.util.Set;

import java.util.concurrent.TimeUnit;

@Component

public class RedisUtil {

@Autowired

private RedisTemplate redisTemplate;

/\*\*

\* 指定缓存失效时间

\*

\* @param key 键

\* @param time 时间(秒)

\* @return

\*/

public boolean expire(String key, long time) {

try {

if (time > 0) {

redisTemplate.expire(key, time, TimeUnit.SECONDS);

}

return true;

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

/\*\*

\* 根据key 获取过期时间

\*

\* @param key 键 不能为null

\* @return 时间(秒) 返回0代表为永久有效

\*/

public long getExpire(String key) {

return redisTemplate.getExpire(key, TimeUnit.SECONDS);

}

/\*\*

\* 判断key是否存在

\*

\* @param key 键

\* @return true 存在 false不存在

\*/

public boolean hasKey(String key) {

try {

return redisTemplate.hasKey(key);

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

/\*\*

\* 删除缓存

\*

\* @param key 可以传一个值 或多个

\*/

@SuppressWarnings("unchecked")

public void del(String... key) {

if (key != null && key.length > 0) {

if (key.length == 1) {

redisTemplate.delete(key[0]);

} else {

redisTemplate.delete(CollectionUtils.arrayToList(key));

}

}

}

/\*\*

\* 普通缓存获取

\*

\* @param key 键

\* @return 值

\*/

public Object get(String key) {

return key == null ? null : redisTemplate.opsForValue().get(key);

}

/\*\*

\* 普通缓存放入

\*

\* @param key 键

\* @param value 值

\* @return true成功 false失败

\*/

public boolean set(String key, Object value) {

try {

redisTemplate.opsForValue().set(key, value);

return true;

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

/\*\*

\* 普通缓存放入并设置时间

\*

\* @param key 键

\* @param value 值

\* @param time 时间(秒) time要大于0 如果time小于等于0 将设置无限期

\* @return true成功 false 失败

\*/

public boolean set(String key, Object value, long time) {

try {

if (time > 0) {

redisTemplate.opsForValue().set(key, value, time, TimeUnit.SECONDS);

} else {

set(key, value);

}

return true;

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

/\*\*

\* 递增

\*

\* @param key 键

\* @param delta 要增加几(大于0)

\* @return

\*/

public long incr(String key, long delta) {

if (delta < 0) {

throw new RuntimeException("递增因子必须大于0");

}

return redisTemplate.opsForValue().increment(key, delta);

}

/\*\*

\* 递减

\*

\* @param key 键

\* @param delta 要减少几(小于0)

\* @return

\*/

public long decr(String key, long delta) {

if (delta < 0) {

throw new RuntimeException("递减因子必须大于0");

}

return redisTemplate.opsForValue().increment(key, -delta);

}

/\*\*

\* HashGet

\*

\* @param key 键 不能为null

\* @param item 项 不能为null

\* @return 值

\*/

public Object hget(String key, String item) {

return redisTemplate.opsForHash().get(key, item);

}

/\*\*

\* 获取hashKey对应的所有键值

\*

\* @param key 键

\* @return 对应的多个键值

\*/

public Map<Object, Object> hmget(String key) {

return redisTemplate.opsForHash().entries(key);

}

/\*\*

\* HashSet

\*

\* @param key 键

\* @param map 对应多个键值

\* @return true 成功 false 失败

\*/

public boolean hmset(String key, Map<String, Object> map) {

try {

redisTemplate.opsForHash().putAll(key, map);

return true;

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

/\*\*

\* HashSet 并设置时间

\*

\* @param key 键

\* @param map 对应多个键值

\* @param time 时间(秒)

\* @return true成功 false失败

\*/

public boolean hmset(String key, Map<String, Object> map, long time) {

try {

redisTemplate.opsForHash().putAll(key, map);

if (time > 0) {

expire(key, time);

}

return true;

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

/\*\*

\* 向一张hash表中放入数据,如果不存在将创建

\*

\* @param key 键

\* @param item 项

\* @param value 值

\* @return true 成功 false失败

\*/

public boolean hset(String key, String item, Object value) {

try {

redisTemplate.opsForHash().put(key, item, value);

return true;

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

/\*\*

\* 向一张hash表中放入数据,如果不存在将创建

\*

\* @param key 键

\* @param item 项

\* @param value 值

\* @param time 时间(秒) 注意:如果已存在的hash表有时间,这里将会替换原有的时间

\* @return true 成功 false失败

\*/

public boolean hset(String key, String item, Object value, long time) {

try {

redisTemplate.opsForHash().put(key, item, value);

if (time > 0) {

expire(key, time);

}

return true;

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

/\*\*

\* 删除hash表中的值

\*

\* @param key 键 不能为null

\* @param item 项 可以使多个 不能为null

\*/

public void hdel(String key, Object... item) {

redisTemplate.opsForHash().delete(key, item);

}

/\*\*

\* 判断hash表中是否有该项的值

\*

\* @param key 键 不能为null

\* @param item 项 不能为null

\* @return true 存在 false不存在

\*/

public boolean hHasKey(String key, String item) {

return redisTemplate.opsForHash().hasKey(key, item);

}

/\*\*

\* hash递增 如果不存在,就会创建一个 并把新增后的值返回

\*

\* @param key 键

\* @param item 项

\* @param by 要增加几(大于0)

\* @return

\*/

public double hincr(String key, String item, double by) {

return redisTemplate.opsForHash().increment(key, item, by);

}

/\*\*

\* hash递减

\*

\* @param key 键

\* @param item 项

\* @param by 要减少记(小于0)

\* @return

\*/

public double hdecr(String key, String item, double by) {

return redisTemplate.opsForHash().increment(key, item, -by);

}

/\*\*

\* 根据key获取Set中的所有值

\*

\* @param key 键

\* @return

\*/

public Set<Object> sGet(String key) {

try {

return redisTemplate.opsForSet().members(key);

} catch (Exception e) {

e.printStackTrace();

return null;

}

}

/\*\*

\* 根据value从一个set中查询,是否存在

\*

\* @param key 键

\* @param value 值

\* @return true 存在 false不存在

\*/

public boolean sHasKey(String key, Object value) {

try {

return redisTemplate.opsForSet().isMember(key, value);

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

/\*\*

\* 将数据放入set缓存

\*

\* @param key 键

\* @param values 值 可以是多个

\* @return 成功个数

\*/

public long sSet(String key, Object... values) {

try {

return redisTemplate.opsForSet().add(key, values);

} catch (Exception e) {

e.printStackTrace();

return 0;

}

}

/\*\*

\* 将set数据放入缓存

\*

\* @param key 键

\* @param time 时间(秒)

\* @param values 值 可以是多个

\* @return 成功个数

\*/

public long sSetAndTime(String key, long time, Object... values) {

try {

Long count = redisTemplate.opsForSet().add(key, values);

if (time > 0) expire(key, time);

return count;

} catch (Exception e) {

e.printStackTrace();

return 0;

}

}

/\*\*

\* 获取set缓存的长度

\*

\* @param key 键

\* @return

\*/

public long sGetSetSize(String key) {

try {

return redisTemplate.opsForSet().size(key);

} catch (Exception e) {

e.printStackTrace();

return 0;

}

}

/\*\*

\* 移除值为value的

\*

\* @param key 键

\* @param values 值 可以是多个

\* @return 移除的个数

\*/

public long setRemove(String key, Object... values) {

try {

Long count = redisTemplate.opsForSet().remove(key, values);

return count;

} catch (Exception e) {

e.printStackTrace();

return 0;

}

}

/\*\*

\* 获取list缓存的内容

\*

\* @param key 键

\* @param start 开始

\* @param end 结束 0 到 -1代表所有值

\* @return

\*/

public List<Object> lGet(String key, long start, long end) {

try {

return redisTemplate.opsForList().range(key, start, end);

} catch (Exception e) {

e.printStackTrace();

return null;

}

}

/\*\*

\* 获取list缓存的长度

\*

\* @param key 键

\* @return

\*/

public long lGetListSize(String key) {

try {

return redisTemplate.opsForList().size(key);

} catch (Exception e) {

e.printStackTrace();

return 0;

}

}

/\*\*

\* 通过索引 获取list中的值

\*

\* @param key 键

\* @param index 索引 index>=0时， 0 表头，1 第二个元素，依次类推；index<0时，-1，表尾，-2倒数第二个元素，依次类推

\* @return

\*/

public Object lGetIndex(String key, long index) {

try {

return redisTemplate.opsForList().index(key, index);

} catch (Exception e) {

e.printStackTrace();

return null;

}

}

/\*\*

\* 将list放入缓存

\*

\* @param key 键

\* @param value 值

\* @return

\*/

public boolean lSet(String key, Object value) {

try {

redisTemplate.opsForList().rightPush(key, value);

return true;

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

/\*\*

\* 将list放入缓存

\*

\* @param key 键

\* @param value 值

\* @param time 时间(秒)

\* @return

\*/

public boolean lSet(String key, Object value, long time) {

try {

redisTemplate.opsForList().rightPush(key, value);

if (time > 0) expire(key, time);

return true;

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

/\*\*

\* 将list放入缓存

\*

\* @param key 键

\* @param value 值

\* @return

\*/

public boolean lSet(String key, List<Object> value) {

try {

redisTemplate.opsForList().rightPushAll(key, value);

return true;

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

/\*\*

\* 将list放入缓存

\*

\* @param key 键

\* @param value 值

\* @param time 时间(秒)

\* @return

\*/

public boolean lSet(String key, List<Object> value, long time) {

try {

redisTemplate.opsForList().rightPushAll(key, value);

if (time > 0) expire(key, time);

return true;

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

/\*\*

\* 根据索引修改list中的某条数据

\*

\* @param key 键

\* @param index 索引

\* @param value 值

\* @return

\*/

public boolean lUpdateIndex(String key, long index, Object value) {

try {

redisTemplate.opsForList().set(key, index, value);

return true;

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

/\*\*

\* 移除N个值为value

\*

\* @param key 键

\* @param count 移除多少个

\* @param value 值

\* @return 移除的个数

\*/

public long lRemove(String key, long count, Object value) {

try {

Long remove = redisTemplate.opsForList().remove(key, count, value);

return remove;

} catch (Exception e) {

e.printStackTrace();

return 0;

}

}

/\*\*

\* 有序set添加元素

\*

\* @param key

\* @param value

\* @param score

\* @return

\*/

public boolean zSet(String key, Object value, double score) {

return redisTemplate.opsForZSet().add(key, value, score);

}

public long batchZSet(String key, Set<ZSetOperations.TypedTuple> typles) {

return redisTemplate.opsForZSet().add(key, typles);

}

public void zIncrementScore(String key, Object value, long delta) {

redisTemplate.opsForZSet().incrementScore(key, value, delta);

}

public void zUnionAndStore(String key, Collection otherKeys, String destKey) {

redisTemplate.opsForZSet().unionAndStore(key, otherKeys, destKey);

}

/\*\*

\* 获取zset数量

\* @param key

\* @param value

\* @return

\*/

public long getZsetScore(String key, Object value) {

Double score = redisTemplate.opsForZSet().score(key, value);

if(score==null){

return 0;

}else{

return score.longValue();

}

}

/\*\*

\* 获取有序集 key 中成员 member 的排名 。

\* 其中有序集成员按 score 值递减 (从大到小) 排序。

\* @param key

\* @param start

\* @param end

\* @return

\*/

public Set<ZSetOperations.TypedTuple> getZSetRank(String key, long start, long end) {

return redisTemplate.opsForZSet().reverseRangeWithScores(key, start, end);

}

}

package fun.redamancyxun.chinese.backend.util;

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

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

import org.springframework.stereotype.Component;

import java.util.concurrent.TimeUnit;

@Component

public class RedisUtilComplex {

@Autowired

private RedisTemplate redisTemplate;

@Autowired

private RedisUtil redisUtil;

public boolean set(String key, Object value, Long timeMillis){

Long expire\_time = timeMillis - System.currentTimeMillis();

try {

if (expire\_time > 0) {

redisTemplate.opsForValue().set(key, value, expire\_time, TimeUnit.MILLISECONDS);

} else {

redisUtil.set(key, value);

}

return true;

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

}

package fun.redamancyxun.chinese.backend.util;

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

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

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

import org.springframework.stereotype.Component;

import org.springframework.util.CollectionUtils;

import java.util.Collection;

import java.util.List;

import java.util.Map;

import java.util.Set;

import java.util.concurrent.TimeUnit;

@Component

@SuppressWarnings("all")

public class RedisUtils {

@Autowired

private RedisTemplate redisTemplate;

/\*\*

\* 指定缓存失效时间

\*

\* @param key 键

\* @param time 时间(秒)

\* @return

\*/

public boolean expire(String key, long time) {

try {

if (time > 0) {

redisTemplate.expire(key, time, TimeUnit.MINUTES);

}

return true;

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

/\*\*

\* 根据key 获取过期时间

\*

\* @param key 键 不能为null

\* @return 时间(秒) 返回0代表为永久有效

\*/

public long getExpire(String key) {

return redisTemplate.opsForValue().getOperations().getExpire(key);

}

public boolean isExpire(String key){

return getExpire(key) > 1 ? false : true;

}

/\*\*

\* 判断key是否存在

\*

\* @param key 键

\* @return true 存在 false不存在

\*/

public boolean hasKey(String key) {

try {

return redisTemplate.hasKey(key);

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

/\*\*

\* 删除缓存

\*

\* @param key 可以传一个值 或多个

\*/

public void del(String... key) {

if (key != null && key.length > 0) {

if (key.length == 1) {

redisTemplate.delete(key[0]);

} else {

redisTemplate.delete(CollectionUtils.arrayToList(key));

}

}

}

/\*\*

\* 普通缓存获取

\*

\* @param key 键

\* @return 值

\*/

public Object get(String key) {

return key == null ? null : redisTemplate.opsForValue().get(key);

}

/\*\*

\* 普通缓存放入

\*

\* @param key 键

\* @param value 值

\* @return true成功 false失败

\*/

public boolean set(String key, Object value) {

try {

redisTemplate.opsForValue().set(key, value);

return true;

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

/\*\*

\* 普通缓存放入并设置时间

\*

\* @param key 键

\* @param value 值

\* @param time 时间(秒) time要大于0 如果time小于等于0 将设置无限期

\* @return true成功 false 失败

\*/

public boolean set(String key, Object value, long time) {

try {

if (time > 0) {

redisTemplate.opsForValue().set(key, value, time, TimeUnit.SECONDS);

} else {

set(key, value);

}

return true;

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

/\*\*

\* 递增

\*

\* @param key 键

\* @param delta 要增加几(大于0)

\* @return

\*/

public long incr(String key, long delta) {

if (delta < 0) {

throw new RuntimeException("递增因子必须大于0");

}

return redisTemplate.opsForValue().increment(key, delta);

}

/\*\*

\* 递减

\*

\* @param key 键

\* @param delta 要减少几(小于0)

\* @return

\*/

public long decr(String key, long delta) {

if (delta < 0) {

throw new RuntimeException("递减因子必须大于0");

}

return redisTemplate.opsForValue().increment(key, -delta);

}

/\*\*

\* HashGet

\*

\* @param key 键 不能为null

\* @param item 项 不能为null

\* @return 值

\*/

public Object hget(String key, String item) {

return redisTemplate.opsForHash().get(key, item);

}

/\*\*

\* 获取hashKey对应的所有键值

\*

\* @param key 键

\* @return 对应的多个键值

\*/

public Map<Object, Object> hmget(String key) {

return redisTemplate.opsForHash().entries(key);

}

/\*\*

\* HashSet

\*

\* @param key 键

\* @param map 对应多个键值

\* @return true 成功 false 失败

\*/

public boolean hmset(String key, Map<String, Object> map) {

try {

redisTemplate.opsForHash().putAll(key, map);

return true;

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

/\*\*

\* HashSet 并设置时间

\*

\* @param key 键

\* @param map 对应多个键值

\* @param time 时间(秒)

\* @return true成功 false失败

\*/

public boolean hmset(String key, Map<String, Object> map, long time) {

try {

redisTemplate.opsForHash().putAll(key, map);

if (time > 0) {

expire(key, time);

}

return true;

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

/\*\*

\* 向一张hash表中放入数据,如果不存在将创建

\*

\* @param key 键

\* @param item 项

\* @param value 值

\* @return true 成功 false失败

\*/

public boolean hset(String key, String item, Object value) {

try {

redisTemplate.opsForHash().put(key, item, value);

return true;

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

/\*\*

\* 向一张hash表中放入数据,如果不存在将创建

\*

\* @param key 键

\* @param item 项

\* @param value 值

\* @param time 时间(秒) 注意:如果已存在的hash表有时间,这里将会替换原有的时间

\* @return true 成功 false失败

\*/

public boolean hset(String key, String item, Object value, long time) {

try {

redisTemplate.opsForHash().put(key, item, value);

if (time > 0) {

expire(key, time);

}

return true;

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

/\*\*

\* 删除hash表中的值

\*

\* @param key 键 不能为null

\* @param item 项 可以使多个 不能为null

\*/

public void hdel(String key, Object... item) {

redisTemplate.opsForHash().delete(key, item);

}

/\*\*

\* 判断hash表中是否有该项的值

\*

\* @param key 键 不能为null

\* @param item 项 不能为null

\* @return true 存在 false不存在

\*/

public boolean hHasKey(String key, String item) {

return redisTemplate.opsForHash().hasKey(key, item);

}

/\*\*

\* hash递增 如果不存在,就会创建一个 并把新增后的值返回

\*

\* @param key 键

\* @param item 项

\* @param by 要增加几(大于0)

\* @return

\*/

public double hincr(String key, String item, double by) {

return redisTemplate.opsForHash().increment(key, item, by);

}

/\*\*

\* hash递减

\*

\* @param key 键

\* @param item 项

\* @param by 要减少记(小于0)

\* @return

\*/

public double hdecr(String key, String item, double by) {

return redisTemplate.opsForHash().increment(key, item, -by);

}

/\*\*

\* 根据key获取Set中的所有值

\*

\* @param key 键

\* @return

\*/

public Set<Object> sGet(String key) {

try {

return redisTemplate.opsForSet().members(key);

} catch (Exception e) {

e.printStackTrace();

return null;

}

}

/\*\*

\* 根据value从一个set中查询,是否存在

\*

\* @param key 键

\* @param value 值

\* @return true 存在 false不存在

\*/

public boolean sHasKey(String key, Object value) {

try {

return redisTemplate.opsForSet().isMember(key, value);

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

/\*\*

\* 将数据放入set缓存

\*

\* @param key 键

\* @param values 值 可以是多个

\* @return 成功个数

\*/

public long sSet(String key, Object... values) {

try {

return redisTemplate.opsForSet().add(key, values);

} catch (Exception e) {

e.printStackTrace();

return 0;

}

}

/\*\*

\* 将set数据放入缓存

\*

\* @param key 键

\* @param time 时间(秒)

\* @param values 值 可以是多个

\* @return 成功个数

\*/

public long sSetAndTime(String key, long time, Object... values) {

try {

Long count = redisTemplate.opsForSet().add(key, values);

if (time > 0) expire(key, time);

return count;

} catch (Exception e) {

e.printStackTrace();

return 0;

}

}

/\*\*

\* 获取set缓存的长度

\*

\* @param key 键

\* @return

\*/

public long sGetSetSize(String key) {

try {

return redisTemplate.opsForSet().size(key);

} catch (Exception e) {

e.printStackTrace();

return 0;

}

}

/\*\*

\* 移除值为value的

\*

\* @param key 键

\* @param values 值 可以是多个

\* @return 移除的个数

\*/

public long setRemove(String key, Object... values) {

try {

Long count = redisTemplate.opsForSet().remove(key, values);

return count;

} catch (Exception e) {

e.printStackTrace();

return 0;

}

}

/\*\*

\* 获取list缓存的内容

\*

\* @param key 键

\* @param start 开始

\* @param end 结束 0 到 -1代表所有值

\* @return

\*/

public List<Object> lGet(String key, long start, long end) {

try {

return redisTemplate.opsForList().range(key, start, end);

} catch (Exception e) {

e.printStackTrace();

return null;

}

}

/\*\*

\* 获取list缓存的长度

\*

\* @param key 键

\* @return

\*/

public long lGetListSize(String key) {

try {

return redisTemplate.opsForList().size(key);

} catch (Exception e) {

e.printStackTrace();

return 0;

}

}

/\*\*

\* 通过索引 获取list中的值

\*

\* @param key 键

\* @param index 索引 index>=0时， 0 表头，1 第二个元素，依次类推；index<0时，-1，表尾，-2倒数第二个元素，依次类推

\* @return

\*/

public Object lGetIndex(String key, long index) {

try {

return redisTemplate.opsForList().index(key, index);

} catch (Exception e) {

e.printStackTrace();

return null;

}

}

/\*\*

\* 将list放入缓存

\*

\* @param key 键

\* @param value 值

\* @return

\*/

public boolean lSet(String key, Object value) {

try {

redisTemplate.opsForList().rightPush(key, value);

return true;

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

/\*\*

\* 将list放入缓存

\*

\* @param key 键

\* @param value 值

\* @param time 时间(秒)

\* @return

\*/

public boolean lSet(String key, Object value, long time) {

try {

redisTemplate.opsForList().rightPush(key, value);

if (time > 0) expire(key, time);

return true;

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

/\*\*

\* 将list放入缓存

\*

\* @param key 键

\* @param value 值

\* @return

\*/

public boolean lSet(String key, List<Object> value) {

try {

redisTemplate.opsForList().rightPushAll(key, value);

return true;

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

/\*\*

\* 将list放入缓存

\*

\* @param key 键

\* @param value 值

\* @param time 时间(秒)

\* @return

\*/

public boolean lSet(String key, List<Object> value, long time) {

try {

redisTemplate.opsForList().rightPushAll(key, value);

if (time > 0) expire(key, time);

return true;

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

/\*\*

\* 根据索引修改list中的某条数据

\*

\* @param key 键

\* @param index 索引

\* @param value 值

\* @return

\*/

public boolean lUpdateIndex(String key, long index, Object value) {

try {

redisTemplate.opsForList().set(key, index, value);

return true;

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

/\*\*

\* 移除N个值为value

\*

\* @param key 键

\* @param count 移除多少个

\* @param value 值

\* @return 移除的个数

\*/

public long lRemove(String key, long count, Object value) {

try {

Long remove = redisTemplate.opsForList().remove(key, count, value);

return remove;

} catch (Exception e) {

e.printStackTrace();

return 0;

}

}

/\*\*

\* 有序set添加元素

\*

\* @param key

\* @param value

\* @param score

\* @return

\*/

public boolean zSet(String key, Object value, double score) {

return redisTemplate.opsForZSet().add(key, value, score);

}

public long batchZSet(String key, Set<ZSetOperations.TypedTuple> typles) {

return redisTemplate.opsForZSet().add(key, typles);

}

public void zIncrementScore(String key, Object value, long delta) {

redisTemplate.opsForZSet().incrementScore(key, value, delta);

}

public void zUnionAndStore(String key, Collection otherKeys, String destKey) {

redisTemplate.opsForZSet().unionAndStore(key, otherKeys, destKey);

}

/\*\*

\* 获取zset数量

\* @param key

\* @param value

\* @return

\*/

public long getZsetScore(String key, Object value) {

Double score = redisTemplate.opsForZSet().score(key, value);

if(score==null){

return 0;

}else{

return score.longValue();

}

}

/\*\*

\* 获取有序集 key 中成员 member 的排名 。

\* 其中有序集成员按 score 值递减 (从大到小) 排序。

\* 从开始(start)到结束(end)，从排序从高到低的排序集中获取元组的集合

\* 把得分从大到小排序，然后取第start到end名（闭区间）

\* @param key

\* @param start

\* @param end

\* @return

\*/

public Set<ZSetOperations.TypedTuple> getZSetRank(String key, long start, long end) {

return redisTemplate.opsForZSet().reverseRangeWithScores(key, start, end);

}

public boolean tset(String key, Object value, Long timeMillis){

Long expire\_time = timeMillis - System.currentTimeMillis();

try {

if (expire\_time > 0) {

redisTemplate.opsForValue().set(key, value, expire\_time, TimeUnit.MILLISECONDS);

} else {

set(key, value);

}

return true;

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

}

package fun.redamancyxun.chinese.backend.util;

import java.lang.reflect.Field;

/\*\*

\* Created with IntelliJ IDEA.

\* User: zhangrui1

\* Date: 14-4-29

\* Time: 上午11:34

\* Description: 反射工具类

\*/

public class ReflectionUtils {

/\*\* 日志记录对象 \*\*/

/\*\*

\* 利用反射获取指定对象的指定属性

\* @param obj 目标对象

\* @param fieldName 目标属性

\* @return 目标属性的值

\*/

public static Object getFieldValue(Object obj, String fieldName) {

Object result = null;

Field field = getField(obj, fieldName);

if (field != null) {

field.setAccessible(true);

try {

result = field.get(obj);

} catch (IllegalArgumentException e) {

e.printStackTrace();

} catch (IllegalAccessException e) {

e.printStackTrace();

}

}

return result;

}

/\*\*

\* 利用反射获取指定对象里面的指定属性

\* @param obj 目标对象

\* @param fieldName 目标属性

\* @return 目标字段

\*/

private static Field getField(Object obj, String fieldName) {

Field field = null;

for (Class<?> clazz=obj.getClass(); clazz != Object.class; clazz=clazz.getSuperclass()) {

try {

field = clazz.getDeclaredField(fieldName);

break;

} catch (NoSuchFieldException e) {

}

}

return field;

}

/\*\*

\* 利用反射设置指定对象的指定属性为指定的值

\* @param obj 目标对象

\* @param fieldName 目标属性

\* @param fieldValue 目标值

\*/

public static void setFieldValue(Object obj, String fieldName,

String fieldValue) {

Field field = getField(obj, fieldName);

if (field != null) {

try {

field.setAccessible(true);

field.set(obj, fieldValue);

} catch (IllegalArgumentException e) {

e.printStackTrace();

} catch (IllegalAccessException e) {

e.printStackTrace();

}

}

}

}

package fun.redamancyxun.chinese.backend.util;

public class RegexUtil {

public final static String PHONE\_NUMBER = "^[1][3,4,5,7,8][0-9]{9}$";

}

package fun.redamancyxun.chinese.backend.util;

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

import fun.redamancyxun.chinese.backend.dto.SessionData;

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.UserMapper;

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

import org.springframework.stereotype.Component;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.util.Objects;

import java.util.Optional;

import java.util.UUID;

/\*\*

\* @author yannis

\* @version 2020/8/1 18:38

\*/

@Component

public class SessionUtils {

@Autowired

private UserMapper userMapper;

@Autowired

private HttpServletRequest request;

@Autowired

private HttpServletResponse response;

@Autowired

private RedisUtils redisUtil;

public String getUserId() {

return Optional

.ofNullable(getSessionData())

.orElse(new SessionData())

.getId();

}

public String getSessionId(String userId) throws MyException {

Object obj = redisUtil.get(userId);

return (String) obj;

}

public SessionData getSessionData() throws MyException {

String key = request.getHeader("session");

if (Objects.equals(key, "shadow")) {

SessionData sessionData = new SessionData();

sessionData.setId("shadow");

sessionData.setRole(2);

return sessionData;

}

if (Objects.equals(key, "admin")) {

SessionData sessionData = new SessionData();

sessionData.setId("admin");

sessionData.setRole(1);

return sessionData;

}

if (Objects.equals(key, "superAdmin")) {

SessionData sessionData = new SessionData();

sessionData.setId("superAdmin");

sessionData.setRole(0);

return sessionData;

}

if (key == null)

throw new MyException(EnumExceptionType.NEED\_SESSION\_ID);

if (redisUtil.isExpire(key)) {

redisUtil.del(key);

throw new MyException(EnumExceptionType.LOGIN\_HAS\_OVERDUE);

}

return (SessionData) redisUtil.get(key);

}

public void setSessionId(String sessionId) {

response.setHeader(CommonConstants.SESSION, sessionId);

}

public String generateSessionId() {

String sessionId = UUID.randomUUID().toString();

response.setHeader(CommonConstants.SESSION, sessionId);

return sessionId;

}

public void ChangeContentType() {

response.setHeader("Content-Type", "application/json");

}

public void invalidate() {

String sessionId = request.getHeader("session");

String userId = getUserId();

redisUtil.del(userId);

redisUtil.del(sessionId);

request.removeAttribute(CommonConstants.SESSION);

}

public void refreshData(User user) throws MyException {

String sessionId = null;

String userId = null;

if (user == null) {

sessionId = request.getHeader("session");

userId = getUserId();

user = userMapper.selectById(userId);

} else {

userId = user.getId();

sessionId = getSessionId(userId);

if (sessionId == null)

throw new MyException(EnumExceptionType.NEED\_SESSION\_ID);

if (redisUtil.isExpire(sessionId)) {

redisUtil.del(sessionId);

throw new MyException(EnumExceptionType.LOGIN\_HAS\_OVERDUE);

}

}

SessionData sessionData = new SessionData(user);

redisUtil.del(sessionId);

redisUtil.set(sessionId, sessionData, 86400);

redisUtil.del(userId);

redisUtil.set(userId, sessionId, 86400);

}

}

package fun.redamancyxun.chinese.backend.util;

import org.springframework.beans.BeansException;

import org.springframework.context.ApplicationContext;

import org.springframework.context.ApplicationContextAware;

import java.util.Locale;

/\*\*

\* 项目名称:

\* 类名: SpringContextUtil

\* 描述： 获取bean的工具类，可用于在线程里面获取bean

\* 创建人: -.-- .- -. --. --.. .. .- -.

\* 修改备注：

\* 版本：1.0

\*/

public class SpringContextUtil implements ApplicationContextAware {

private static ApplicationContext context = null;

/\* (non Javadoc)

\* @Title: setApplicationContext

\* @Description: spring获取bean工具类

\* @param applicationContext

\* @throws BeansException

\* @see org.springframework.context.ApplicationContextAware#setApplicationContext(org.springframework.context.ApplicationContext)

\*/

@Override

public void setApplicationContext(ApplicationContext applicationContext)

throws BeansException {

context = applicationContext;

}

public static <T> T getBean(String beanName){

return (T) context.getBean(beanName);

}

public static String getMessage(String key){

return context.getMessage(key, null, Locale.getDefault());

}

}

package fun.redamancyxun.chinese.backend.util;

import org.springframework.web.servlet.HandlerInterceptor;

import org.springframework.web.servlet.ModelAndView;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

public class SpringInterceptor implements HandlerInterceptor {

@Override

public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws Exception {

response.setContentType("application/json");

return true;

}

@Override

public void postHandle(HttpServletRequest request, HttpServletResponse response, Object handler, ModelAndView modelAndView) throws Exception {

}

@Override

public void afterCompletion(HttpServletRequest request, HttpServletResponse response, Object handler, Exception ex) throws Exception {

}

}

package fun.redamancyxun.chinese.backend.util;

import org.springframework.stereotype.Component;

import java.sql.Timestamp;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

/\*\*

\* @author yannis

\* @version 2020/11/7 9:40

\*/

@Component

public class TimeUtil {

public static String getCurrentTimestamp() {

String time = new Timestamp(System.currentTimeMillis()).toString();

time = time.substring(0, 19);

return time;

}

public static String getCurrentTimestampForWxSend(String time) {

time = time.substring(0, 16);

StringBuilder sb = new StringBuilder(time);

sb.replace(4, 5, "年");

sb.replace(7, 8, "月");

sb.replace(10, 10, "日");

return sb.toString();

}

public static Date parseToDate(String timeStamp) {

SimpleDateFormat mDateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

Date date = null;

try {

date = mDateFormat.parse(timeStamp);

} catch (ParseException e) {

e.printStackTrace();

}

return date;

}

public static void main(String[] args) {

System.out.println(getCurrentTimestampForWxSend(getCurrentTimestamp()));

}

}

package fun.redamancyxun.chinese.backend.util;

import net.sf.jsqlparser.expression.Expression;

import net.sf.jsqlparser.expression.LongValue;

import net.sf.jsqlparser.expression.operators.arithmetic.Addition;

import net.sf.jsqlparser.expression.operators.conditional.AndExpression;

import net.sf.jsqlparser.expression.operators.relational.EqualsTo;

import net.sf.jsqlparser.parser.CCJSqlParserUtil;

import net.sf.jsqlparser.schema.Column;

import net.sf.jsqlparser.statement.Statement;

import net.sf.jsqlparser.statement.update.Update;

import org.apache.ibatis.executor.statement.StatementHandler;

import org.apache.ibatis.mapping.BoundSql;

import org.apache.ibatis.mapping.MappedStatement;

import org.apache.ibatis.mapping.SqlCommandType;

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

import org.apache.ibatis.reflection.MetaObject;

import org.apache.ibatis.reflection.SystemMetaObject;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import java.lang.reflect.Proxy;

import java.sql.Connection;

import java.util.List;

import java.util.Properties;

/\*\*

\* 乐观锁：数据版本插件

\*

\*/

@Intercepts(

@Signature(

type = StatementHandler.class,

method = "prepare",

args = {Connection.class, Integer.class}

)

)

public class VersionInterceptor implements Interceptor {

private static final String VERSION\_COLUMN\_NAME = "version";

private static final Logger logger = LoggerFactory.getLogger(VersionInterceptor.class);

@Override

public Object intercept(Invocation invocation) throws Throwable {

StatementHandler handler = (StatementHandler) processTarget(invocation.getTarget());

MetaObject metaObject = SystemMetaObject.forObject(handler);

MappedStatement ms = (MappedStatement) metaObject.getValue("delegate.mappedStatement");

SqlCommandType sqlType = ms.getSqlCommandType();

if(sqlType != SqlCommandType.UPDATE) {

return invocation.proceed();

}

Object originalVersion = metaObject.getValue("delegate.boundSql.parameterObject." + VERSION\_COLUMN\_NAME);

if(originalVersion == null || Long.parseLong(originalVersion.toString()) <= 0){

return invocation.proceed();

}

BoundSql boundSql = (BoundSql) metaObject.getValue("delegate.boundSql");

String originalSql = boundSql.getSql();

originalSql = addVersionToSql(originalSql, originalVersion);

metaObject.setValue("delegate.boundSql.sql", originalSql);

return invocation.proceed();

}

/\*\*

\* Plugin.wrap 方法会自动判断拦截器的签名和被拦截对象的接口是否匹配，只有匹配的情况下才会使用动态代理拦截目标对象.

\*

\* @param target 被拦截的对象

\* @return 代理对象

\*/

@Override

public Object plugin(Object target) {

return Plugin.wrap(target, this);

}

/\*\*

\* 设置参数

\*/

@Override

public void setProperties(Properties properties) {

}

/\*\*

\* 获取代理的原始对象

\*

\* @param target

\* @return

\*/

private static Object processTarget(Object target) {

if(Proxy.isProxyClass(target.getClass())) {

MetaObject mo = SystemMetaObject.forObject(target);

return processTarget(mo.getValue("h.target"));

}

return target;

}

/\*\*

\* 为原SQL添加version

\*

\* @param originalSql 原SQL

\* @param originalVersion 原版本号

\* @return 加入version的SQL

\*/

private String addVersionToSql(String originalSql, Object originalVersion){

try{

Statement stmt = CCJSqlParserUtil.parse(originalSql);

if(!(stmt instanceof Update)){

return originalSql;

}

Update update = (Update)stmt;

if(contains(update)){

buildVersionExpression(update);

}

Expression where = update.getWhere();

if(where != null){

AndExpression and = new AndExpression(where, buildVersionEquals(originalVersion));

update.setWhere(and);

}else{

update.setWhere(buildVersionEquals(originalVersion));

}

return stmt.toString();

}catch(Exception e){

logger.error(e.getMessage(), e);

return originalSql;

}

}

private boolean contains(Update update){

List<Column> columns = update.getColumns();

for(Column column : columns){

if(column.getColumnName().equalsIgnoreCase(VERSION\_COLUMN\_NAME)){

return true;

}

}

return false;

}

private void buildVersionExpression(Update update){

Column versionColumn = new Column();

versionColumn.setColumnName(VERSION\_COLUMN\_NAME);

update.getColumns().add(versionColumn);

Addition add = new Addition();

add.setLeftExpression(versionColumn);

add.setRightExpression(new LongValue(1));

update.getExpressions().add(add);

}

private Expression buildVersionEquals(Object originalVersion){

Column column = new Column();

column.setColumnName(VERSION\_COLUMN\_NAME);

EqualsTo equal = new EqualsTo();

equal.setLeftExpression(column);

equal.setRightExpression(new LongValue(originalVersion.toString()));

return equal;

}

}

server:

port: 8080 #HTTPS协议默认端口号为443，需要使用其他端口时，您可以在此处自定义。

ssl:

key-store-password: s8h1qc0x #填写jks-password.txt文件内的密码。

key-store-type: JKS #JKS格式证书密钥库类型。PFX格式的证书密钥库类型为PKCS12。

key-store: classpath:chinese.redamancyxun.fun.jks #您需要使用实际的证书名称替换domain\_name.jks。

tomcat:

uriEncoding: UTF-8

servlet:

encoding:

force: true

charset: UTF-8

enabled: true

spring:

#?????????QQ?????????

servlet:

multipart:

enabled: true

max-file-size: 10MB

max-request-size: 10MB

mail:

host: smtp.qq.com #???QQ?????????? SMTP ?????, ????, ???????????, ??(????, ????)???: smtp.xxx.com?????

port: 587

username: 2044978721@qq.com #qq??

password: pajdcdkinctsbchd #qq?????

protocol: smtp #??????

properties.mail.smtp.auth: true #????????????true,?????????????

properties.mail.smtp.starttls.enable: true

properties.mail.smtp.starttls.required: true

properties.mail.smtp.ssl.enable: false #??SSL ???????

default-encoding: utf-8

datasource:

url: jdbc:mysql://139.196.124.95:3306/chinese\_scoring\_model?useUnicode=true&characterEncoding=utf-8&useSSL=false&serverTimezone=Asia/Shanghai&rewriteBatchedStatements=true&allowPublicKeyRetrieval=true

driver-class-name: com.mysql.cj.jdbc.Driver

username: root

password: a13738780767

jackson:

time-zone: GMT+8

jpa:

hibernate:

ddl-auto: update

show-sql: true

data:

redis:

database: 0

host: 139.196.124.95

port: 6379

timeout: 1000

# password: fu040913

mybatis-plus:

global-config:

db-config:

#????????NULL??

field-strategy: not\_null

configuration:

map-underscore-to-camel-case: true

logback:

dir: logs

#????

filename:

all: springboot-base-frame-all

info: springboot-base-frame-info

warn: springboot-base-frame-warn

error: springboot-base-frame-error

#???????

maxhistory: 10

#?????????

maxfilesize: 100MB

#????

level: debug

#??????

charset: UTF-8

#????

CONSOLE\_LOG\_PATTERN: '%d{yyyy-MM-dd HH:mm:ss.SSS} %boldYellow([%thread]) %highlight(%-5level) %boldGreen(%logger{50}) - %msg%n'

FILE\_LOG\_PATTERN: '%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger{50} - %msg%n'

uniapp:

appid: wxa70d503b20ed975b #??????????????appid????????

appsecret: fad288a10751b57c04d064a0f2e1edac #??????????????appsecret????????