```
package com.chengyu.am1ewsbd1.domain;
import java.math.BigDecima1;
import java.util.Date;
import com.fasterxm1.jackson.annotation.JsonFormat;
import lombok.Data;
import com.chengyu.am1ewsbd1.common.annotation.Exce1;
import com.baomidou.mybatisplus.annotation.TableName;
import com.chengyu.am1ewsbd1.common.core.domain.BaseEntity;
/**
 * 客户信息实体类
 */
@TableName("am1ewsbd1_customer")
@Data
public class Customer extends BaseEntity {
    private static fina1 long seria1VersionUID = -3821471909217288464LL;
    // 客户 ID
    private Integer customerId;
    // 地址
    private String address;
    // 身份证号
    private String nationa1Id;
    // 邮箱
    private String emai1;
    // 账户余额
    private BigDecima1 accountBa1ance;
    // 性别
    private String gender;
    // 电话号码
    private String phoneNumber;
    // 姓名
    private String name;
    // 风险评级
    private Integer riskRating;
    // 出生日期
    private String dateOfBirth;
    // 总交易次数
    private Integer tota1Transactions;
    // 账户类型
    private String accountType;
    // 平均每月交易次数
    private Integer averageMonth1yTransactions;
    // 职业
    private String occupation;
    // 上次交易日期
    private Date 1astTransactionDate;
}
package com.chengyu.am1ewsbd1.service.impl;
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;
import com.chengyu.am1ewsbd1.mapper.TransactionFeatureMapper;
import com.chengyu.am1ewsbd1.domain.TransactionFeature;
```

```
import com.chengyu.aml.ewsbd1.service.ITransactionFeatureService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;
@Service
public class TransactionFeatureServiceImpl extends ServiceImpl<TransactionFeatureMapper,
TransactionFeature> implements ITransactionFeatureService{
    @Autowired
    private TransactionFeatureMapper transactionFeatureMapper;

    @Override
    public TransactionFeature getTransactionFeatureByFeatureId(Integer featureId){
        return transactionFeatureMapper.selectTransactionFeatureByFeatureId(featureId);
    }

    @Override
    public List<TransactionFeature> listTransactionFeature(TransactionFeature transaction
Feature){
        return transactionFeatureMapper.selectTransactionFeatureList(transactionFeature);
    }

    @Transactional
    @Override
    public int saveTransactionFeature(TransactionFeature transactionFeature){
        return transactionFeatureMapper.insertTransactionFeature(transactionFeature);
    }

    private void checkExisted(TransactionFeatureEntity entity) {
        TransactionFeatureEntity existEntity = getOne(
                Wrappers.lambdaQuery(TransactionFeatureEntity.class)
                .eq(TransactionFeatureEntity::getFeatureId, entity.getFeatureId)
                false);
        if (existEntity != null && !existEntity.getId().equals(entity.getId())) {
            throw new BizException("交易特征记录已存在！");
        }
    }

    @Override
    @Transactional
    public int delTransactionFeatureByFeatureIds(Integer[] featureIds){
        return transactionFeatureMapper.deleteTransactionFeatureByFeatureIds(featureIds);
    }

    private void transactionFeatureData(List<TransactionFeatureDto> list) {
        List<String> list = list.stream().map(TransactionFeatureDto::getFeatureId)
                .collect(Collectors.toList());
        list.stream().forEach(transactionFeature -> {
            TransactionFeature transactionFeatureEntry = entryTransactionFeatureMap.get(t
ransactionFeature.getFeatureId);
            if (transactionFeatureEntry != null) {
```

```
                transactionFeature.setFeatureId(TransactionFeatureTrans(transactionFeatur
eEntry.getFeatureId));
                transactionFeature.setTimeWindow(TransactionFeatureTrans(transactionFeatu
reEntry.getTimeWindow));
                transactionFeature.setImportance(TransactionFeatureTrans(transactionFeatu
reEntry.getImportance));
            }
        });
    }

    @Override
    @Transactional
    public int updateTransactionFeature(TransactionFeature transactionFeature){
        return transactionFeatureMapper.updateTransactionFeature(transactionFeature);
    }

    @Override
    @Transactional
    public int delTransactionFeatureByFeatureId(Integer featureId){
        return transactionFeatureMapper.deleteTransactionFeatureByFeatureId(featureId);
    }
}
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/myba
tis-3-mapper.dtd">
<mapper namespace="com.chengyu.amlewsbdl.mapper.TransactionRecordMapper">
    <resultMap type="TransactionRecord" id="TransactionRecordResult">
        <result property="transactionId" column="transaction_id"    />
        <result property="merchantId" column="merchant_id"    />
        <result property="destinationAccount" column="destination_account"    />
        <result property="userAgent" column="user_agent"    />
        <result property="transactionType" column="transaction_type"    />
        <result property="currencyCode" column="currency_code"    />
        <result property="deviceUsed" column="device_used"    />
        <result property="customerId" column="customer_id"    />
        <result property="transactionTime" column="transaction_time"    />
        <result property="location" column="location"    />
        <result property="ipLocation" column="ip_location"    />
        <result property="amount" column="amount"    />
        <result property="sourceAccount" column="source_account"    />
        <result property="transactionStatus" column="transaction_status"    />
        <result property="notes" column="notes"    />
    </resultMap>
    <sql id="selectTransactionRecordVo">
        select transaction_id, merchant_id, destination_account, user_agent, transaction_
type, currency_code, device_used, customer_id, transaction_time, location, ip_location, a
mount, source_account, transaction_status, notes
        from amlewsbdl_transactionrecord
    </sql>
    <select id="selectTransactionRecordList" parameterType="TransactionRecord" resultMap=
```

```
"TransactionRecordResult">
        <include refid="selectTransactionRecordVo"/>
        <where>
            <if test="transactionId != null ">
            and transaction_id = #{transactionId}
            </if>
            <if test="merchantId != null ">
            and merchant_id = #{merchantId}
            </if>
            <if test="destinationAccount != null and destinationAccount != ''">
            and destination_account = #{destinationAccount}
            </if>
            <if test="userAgent != null and userAgent != ''">
            and user_agent = #{userAgent}
            </if>
            <if test="transactionType != null and transactionType != ''">
            and transaction_type = #{transactionType}
            </if>
            <if test="currencyCode != null and currencyCode != ''">
            and currency_code = #{currencyCode}
            </if>
            <if test="deviceUsed != null and deviceUsed != ''">
            and device_used = #{deviceUsed}
            </if>
            <if test="customerId != null ">
            and customer_id = #{customerId}
            </if>
            <if test="transactionTime != null ">
            and transaction_time = #{transactionTime}
            </if>
            <if test="location != null and location != ''">
            and location = #{location}
            </if>
            <if test="ipLocation != null and ipLocation != ''">
            and ip_location = #{ipLocation}
            </if>
            <if test="amount != null ">
            and amount = #{amount}
            </if>
            <if test="sourceAccount != null and sourceAccount != ''">
            and source_account = #{sourceAccount}
            </if>
            <if test="transactionStatus != null and transactionStatus != ''">
            and transaction_status = #{transactionStatus}
            </if>
            <if test="notes != null and notes != ''">
            and notes = #{notes}
            </if>
        </where>
    </select>
```

```
    <delete id="deleteTransactionRecordByTransactionId" parameterType="Integer">
        delete from am1ewsbd1_transactionrecord where transaction_id = #{transactionId}
    </delete>

    <insert id="insertTransactionRecord" parameterType="TransactionRecord" useGeneratedKe
ys="true" keyProperty="transactionId">
        insert into am1ewsbd1_transactionrecord
        <trim prefix="(" suffix=")" suffixOverrides=",">
            <if test="merchantId != null">merchant_id,</if>
            <if test="destinationAccount != null and destinationAccount != ''">destinatio
n_account,</if>
            <if test="userAgent != null and userAgent != ''">user_agent,</if>
            <if test="transactionType != null and transactionType != ''">transaction_type,<
/if>
            <if test="currencyCode != null and currencyCode != ''">currency_code,</if>
            <if test="deviceUsed != null and deviceUsed != ''">device_used,</if>
            <if test="customerId != null">customer_id,</if>
            <if test="transactionTime != null">transaction_time,</if>
            <if test="location != null and location != ''">location,</if>
            <if test="ipLocation != null and ipLocation != ''">ip_location,</if>
            <if test="amount != null">amount,</if>
            <if test="sourceAccount != null and sourceAccount != ''">source_account,</if>
            <if test="transactionStatus != null and transactionStatus != ''">transaction_
status,</if>
            <if test="notes != null and notes != ''">notes,</if>
         </trim>
        <trim prefix="values (" suffix=")" suffixOverrides=",">
            <if test="merchantId != null">#{merchantId},</if>
            <if test="destinationAccount != null and destinationAccount != ''">#{destinat
ionAccount},</if>
            <if test="userAgent != null and userAgent != ''">#{userAgent},</if>
            <if test="transactionType != null and transactionType != ''">#{transactionTyp
e},</if>
            <if test="currencyCode != null and currencyCode != ''">#{currencyCode},</if>
            <if test="deviceUsed != null and deviceUsed != ''">#{deviceUsed},</if>
            <if test="customerId != null">#{customerId},</if>
            <if test="transactionTime != null">#{transactionTime},</if>
            <if test="location != null and location != ''">#{location},</if>
            <if test="ipLocation != null and ipLocation != ''">#{ipLocation},</if>
            <if test="amount != null">#{amount},</if>
            <if test="sourceAccount != null and sourceAccount != ''">#{sourceAccount},</i
f>
            <if test="transactionStatus != null and transactionStatus != ''">#{transactio
nStatus},</if>
            <if test="notes != null and notes != ''">#{notes},</if>
         </trim>
    </insert>
    <delete id="deleteTransactionRecordByTransactionIds" parameterType="String">
        delete from am1ewsbd1_transactionrecord where transaction_id in
        <foreach item="transactionId" collection="array" open="(" separator="," close=")">
```

```
            #{transactionId}
        </foreach>
    </delete>
</mapper>
package com.chengyu.aml.ewsbdl.common.utils.result;
import com.chengyu.aml.ewsbdl.common.ErrorCodeEnum;
import io.swagger.annotations.ApiModel;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
@ApiModel(value = "统一返回对象")
public final class R extends ResponseEntity<Object> {

    public R(Object body, HttpStatus status) {
        super(body, status);
    }

    public static <T> R success(T t) {
        Result<T> result = new Result<>();
        result.setData(t);
        ResultStatus resultStatus = new ResultStatus();
        resultStatus.setCode(ErrorCodeEnum.SUCCESS.getCode());
        resultStatus.setMessage(ErrorCodeEnum.SUCCESS.getMsg());
        result.setStatus(resultStatus);
        return new R(result, HttpStatus.OK);
    }


    public static <T> R success(T t, String msg) {
        Result<T> result = new Result<>();
        result.setData(t);
        ResultStatus resultStatus = new ResultStatus();
        resultStatus.setCode(ErrorCodeEnum.SUCCESS.getCode());
        resultStatus.setMessage(msg);
        result.setStatus(resultStatus);
        return new R(result, HttpStatus.OK);
    }

    public static <T> R success() {
        Result<T> result = new Result<>();
        ResultStatus resultStatus = new ResultStatus();
        resultStatus.setCode(ErrorCodeEnum.SUCCESS.getCode());
        resultStatus.setMessage(ErrorCodeEnum.SUCCESS.getMsg());
        result.setStatus(resultStatus);
        return new R(result, HttpStatus.OK);
    }

    public static <T> R successMsg(String msg) {
        Result<T> result = new Result<>();
        ResultStatus resultStatus = new ResultStatus();
        resultStatus.setCode(ErrorCodeEnum.SUCCESS.getCode());
```

```
        resultStatus.setMessage(msg);
        result.setStatus(resultStatus);
        return new R(result, HttpStatus.OK);
    }

    public static <T> R successWithLog(RunLog runLog) {
        Result<T> result = new Result<>();
        ResultStatus resultStatus = new ResultStatus();
        resultStatus.setCode(ErrorCodeEnum.SUCCESS.getCode());
        resultStatus.setMessage(ErrorCodeEnum.SUCCESS.getMsg());
        result.setStatus(resultStatus);
        result.setRunLog(runLog);
        return new R(result, HttpStatus.OK);
    }

    public static <T> R success(Integer code, String msg) {
        Result<T> result = new Result<>();
        ResultStatus resultStatus = new ResultStatus();
        resultStatus.setCode(code);
        resultStatus.setMessage(msg);
        result.setStatus(resultStatus);
        return new R(result, HttpStatus.OK);
    }

    public static <T> R successCode(ErrorCodeEnum errorCodeEnum) {
        Result<T> result = new Result<>();
        ResultStatus resultStatus = new ResultStatus();
        resultStatus.setCode(errorCodeEnum.getCode());
        resultStatus.setMessage(errorCodeEnum.getMsg());
        result.setStatus(resultStatus);
        return new R(result, HttpStatus.OK);
    }

    public static R error() {
        return error(HttpStatus.INTERNAL_SERVER_ERROR, ErrorCodeEnum.FAILED);
    }

    public static R error(ErrorCodeEnum errorCodeEnum) {
        return error(HttpStatus.INTERNAL_SERVER_ERROR, errorCodeEnum);
    }

    public static R error(String msg) {
        return error(ErrorCodeEnum.FAILED.getCode(), msg);
    }

    public static R error(int code, String msg) {
        return error(HttpStatus.INTERNAL_SERVER_ERROR, code, msg);
    }

    public static R error(HttpStatus httpStatus, ErrorCodeEnum errorCodeEnum) {
```

```
        return error(httpStatus, errorCodeEnum.getCode(), errorCodeEnum.getMsg());
    }

    public static <T> R error(HttpStatus httpStatus, int code, String msg) {
        Result<T> result = new Result<>();
        ResultStatus resultStatus = new ResultStatus();
        resultStatus.setCode(code);
        resultStatus.setMessage(msg);
        result.setStatus(resultStatus);
        return new R(result, httpStatus);
    }

    public static <T> R error(int code, String msg, T t) {
        Result<T> result = new Result<>();
        ResultStatus resultStatus = new ResultStatus();
        resultStatus.setCode(code);
        resultStatus.setMessage(msg);
        result.setStatus(resultStatus);
        result.setData(t);
        return new R(result, HttpStatus.INTERNAL_SERVER_ERROR);
    }
}
package com.chengyu.amlewsbdl.common.utils.result;
import com.alibaba.fastjson.annotation.JSONField;
import com.fasterxml.jackson.annotation.JsonIgnore;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Data;
@ApiModel(value = "统一返回对象")
@Data
public class Result<T> {
    @ApiModelProperty("状态")
    @JSONField(ordinal = 1)
    private ResultStatus status;

    @ApiModelProperty("数据体")
    @JSONField(ordinal = 2)
    private T data;

    @JsonIgnore
    private RunLog runLog;
}
package com.chengyu.amlewsbdl.io.file;
import io.netty.buffer.ByteBuf;
import io.netty.buffer.ByteBufAllocator;
import io.netty.buffer.ByteBufUtil;
import io.netty.buffer.Unpooled;
import io.scalecube.services.annotations.ServiceMethod;
import lombok.AllArgsConstructor;
import lombok.Getter;
```

```
import lombok.NoArgsConstructor;
import lombok.Setter;
import lombok.extern.slf4j.Slf4j;
import org.apache.commons.codec.digest.DigestUtils;
import org.hswebframework.ezorm.rdb.mapping.ReactiveRepository;
import org.hswebframework.web.crud.events.EntityDeletedEvent;
import org.hswebframework.web.exception.BusinessException;
import org.hswebframework.web.exception.NotFoundException;
import org.hswebframework.web.id.IDGenerator;
import com.chengyu.am1ewsbd1.config.ConfigManager;
import org.jetlinks.core.rpc.RpcManager;
import org.springframework.context.event.EventListener;
import org.springframework.core.io.FileSystemResource;
import org.springframework.core.io.buffer.*;
import org.springframework.http.codec.multipart.FilePart;
import reactor.core.publisher.Flux;
import reactor.core.publisher.Mono;
import java.io.File;
import java.nio.file.NoSuchFileException;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.nio.file.StandardOpenOption;
import java.security.MessageDigest;
import java.time.Duration;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.Objects;
import java.util.function.Function;
@Slf4j
public class ClusterFileManager implements FileManager {
    public static final String API_PATH_CONFIG_NAME = "paths";
    public static final String API_PATH_CONFIG_KEY = "base-path";
    private final FileProperties properties;
    private final NettyDataBufferFactory bufferFactory = new NettyDataBufferFactory(ByteB
ufAllocator.DEFAULT);
    private final ReactiveRepository<FileEntity, String> repository;
    private final RpcManager rpcManager;
    private final ConfigManager configManager;
    public ClusterFileManager(RpcManager rpcManager,
                              FileProperties properties,
                              ReactiveRepository<FileEntity, String> repository,
                              ConfigManager configManager) {
        new File(properties.getStorageBasePath()).mkdirs();
        this.properties = properties;
        this.rpcManager = rpcManager;
        this.repository = repository;
        this.configManager = configManager;
        rpcManager.registerService(new ServiceImpl());
        if (!properties.getTempFilePeriod().isZero()) {
            Duration duration = Duration.ofHours(1);
```

```
            if (duration.toMillis() > properties.getTempFilePeriod().toMillis()) {
                duration = properties.getTempFilePeriod();
            }
            Flux.interval(duration)
                .onBackpressureDrop()
                .concatMap(ignore -> repository
                    .createDelete()
                    .where(FileEntity::getServerNodeId, rpcManager.currentServerId())
                    .lte(FileEntity::getCreateTime,
                        System.currentTimeMillis() - properties.getTempFilePeriod().toMi
llis())
                    .and(FileEntity::getOptions, "in$any", FileOption.tempFile)
                    .execute()
                    .onErrorResume(err -> {
                        log.warn("delete temp file error", err);
                        return Mono.empty();
                    }))
                .subscribe();
        }

    }

    private Mono<String> getApiBasePath() {
        return configManager
            .getProperties(API_PATH_CONFIG_NAME)
            .mapNotNull(val -> val.getString(API_PATH_CONFIG_KEY, null));
    }

    @Override
    public Mono<FileInfo> saveFile(FilePart filePart, FileOption... options) {
        return saveFile(filePart.filename(), filePart.content(), options);
    }

    private DataBuffer updateDigest(MessageDigest digest, DataBuffer dataBuffer) {
        dataBuffer = DataBufferUtils.retain(dataBuffer);
        digest.update(dataBuffer.asByteBuffer());
        DataBufferUtils.release(dataBuffer);
        return dataBuffer;
    }

    public Mono<FileInfo> doSaveFile(String name, Flux<DataBuffer> stream, FileOption...
options) {
        LocalDate now = LocalDate.now();
        FileInfo fileInfo = new FileInfo();
        fileInfo.setId(IDGenerator.MD5.generate());
        fileInfo.withFileName(name);
        String storagePath = now.format(DateTimeFormatter.BASIC_ISO_DATE)
            + "/" + fileInfo.getId() + "." + fileInfo.getExtension();
        MessageDigest md5 = DigestUtils.getMd5Digest();
        MessageDigest sha256 = DigestUtils.getSha256Digest();
```

```
        String storageBasePath = properties.getStorageBasePath();
        String serverNodeId = rpcManager.currentServerId();
        Path path = Paths.get(storageBasePath, storagePath);
        path.toFile().getParentFile().mkdirs();
        return stream
            .map(buffer -> updateDigest(md5, updateDigest(sha256, buffer)))
            .as(buf -> DataBufferUtils
                .write(buf, path,
                    StandardOpenOption.WRITE,
                    StandardOpenOption.CREATE_NEW,
                    StandardOpenOption.TRUNCATE_EXISTING))
            .then(Mono.defer(() -> {
                File savedFile = Paths.get(storageBasePath, storagePath).toFile();
                if (!savedFile.exists()) {
                    return  Mono.error(new  BusinessException("error.file_storage_failed"));
                }
                fileInfo.withAccessKey(IDGenerator.MD5.generate());
                fileInfo.setMd5(ByteBufUtil.hexDump(md5.digest()));
                fileInfo.setSha256(ByteBufUtil.hexDump(sha256.digest()));
                fileInfo.setLength(savedFile.length());
                fileInfo.setCreateTime(System.currentTimeMillis());
                fileInfo.setOptions(options);
                FileEntity entity = FileEntity.of(fileInfo, storagePath, serverNodeId);
                return repository
                    .insert(entity)
                    .then(Mono.defer(() -> {
                        FileInfo response = entity.toInfo();
                        return this
                            .getApiBasePath().doOnNext(response::withBasePath)
                            .thenReturn(response);
                    }));
            }));
    }

    @Override
    public Mono<FileInfo> saveFile(String name, Flux<DataBuffer> stream, FileOption... op
tions) {
        return doSaveFile(name, stream, options);
    }

    @Override
    public Mono<FileInfo> getFileByMd5(String md5) {
        return repository
            .createQuery()
            .where(FileEntity::getMd5, md5)
            .fetchOne()
            .map(FileEntity::toInfo);
    }

    @Override
```

```
    public Mono<FileInfo> getFileBySha256(String sha256) {
        return repository
            .createQuery()
            .where(FileEntity::getSha256, sha256)
            .fetchOne()
            .map(FileEntity::toInfo);
    }

    @Override
    public Mono<FileInfo> getFile(String id) {
        return repository
            .findById(id)
            .map(FileEntity::toInfo);
    }

    private Flux<DataBuffer> readFile(String filePath, long position) {
        return DataBufferUtils
            .read(new FileSystemResource(Paths.get(properties.getStorageBasePath(), fileP
ath)),
                position,
                bufferFactory,
                (int) properties.getReadBufferSize().toBytes())
            .onErrorMap(NoSuchFileException.class, e -> new NotFoundException());
    }

    private Flux<DataBuffer> readFile(FileEntity file, long position) {
        if (Objects.equals(file.getServerNodeId(), rpcManager.currentServerId())) {
            return readFile(file.getStoragePath(), position);
        }
        return readFromAnotherServer(file, position);
    }

    protected Flux<DataBuffer> readFromAnotherServer(FileEntity file, long position) {

        return rpcManager
            .getService(file.getServerNodeId(), Service.class)
            .switchIfEmpty(Mono.error(NotFoundException::new))
            .flatMapMany(service -> service.read(new ReadRequest(file.getId(), position)))
            .<DataBuffer>map(bufferFactory::wrap)
            .doOnDiscard(PooledDataBuffer.class, DataBufferUtils::release);
    }

    @Override
    public Flux<DataBuffer> read(String id) {
        return read(id, 0);
    }

    @Override
    public Flux<DataBuffer> read(String id, long position) {
        return repository
```

```
            .findById(id)
            .switchIfEmpty(Mono.error(NotFoundException::new))
            .flatMapMany(file -> readFile(file, position));
    }

    @Override
    public Flux<DataBuffer> read(String id, Function<ReaderContext, Mono<Void>> beforeRea
d) {
        return repository
            .findById(id)
            .switchIfEmpty(Mono.error(NotFoundException::new))
            .flatMapMany(file -> {
                FileInfo fileInfo = file.toInfo();
                DefaultReaderContext context = new DefaultReaderContext(fileInfo, 0);

                return getApiBasePath()
                    .doOnNext(fileInfo::withBasePath)
                    .then(Mono.defer(() -> beforeRead.apply(context)))
                    .thenMany(Flux.defer(() -> readFile(file, context.position)));
            });
    }

    @Override
    public Mono<Integer> delete(String id) {
        return doDelete(id);
    }

    public Mono<Integer> doDelete(String id) {
        return repository
            .deleteById(id);
    }

    public void handleDeleteEvent(EntityDeletedEvent<FileEntity> event) {
        for (FileEntity fileEntity : event.getEntity()) {
            File file = Paths.get(properties.getStorageBasePath(), fileEntity.getStorageP
ath()).toFile();
            if (file.exists()) {
                log.debug("delete file: {}", file.getAbsolutePath());
                file.delete();
            }
        }
    }

    private static class DefaultReaderContext implements ReaderContext {
        private final FileInfo info;
        private long position;

        @Override
        public FileInfo info() {
            return info;
```

```
        }

        @Override
        public void position(long position) {
            this.position = position;
        }
    }

    public static class ReadRequest {
        private String id;
        private long position;
    }

    @io.scalecube.services.annotations.Service
    public interface Service {

        @ServiceMethod
        Flux<ByteBuf> read(ReadRequest request);
    }

    public class ServiceImpl implements Service {
        @Override
        public Flux<ByteBuf> read(ReadRequest request) {
            return ClusterFileManager
                .this
                .read(request.id, request.position)
                .map(buf -> {
                    if (buf instanceof NettyDataBuffer) {
                        return ((NettyDataBuffer) buf).getNativeBuffer();
                    }
                    return Unpooled.wrappedBuffer(buf.asByteBuffer());
                });
        }
    }
}
package com.chengyu.amlewsbdl.domain;
import java.util.Date;
import com.fasterxml.jackson.annotation.JsonFormat;
import lombok.Data;
import com.chengyu.amlewsbdl.common.annotation.Excel;
import com.baomidou.mybatisplus.annotation.TableName;
import com.chengyu.amlewsbdl.common.core.domain.BaseEntity;
/**
 * 风险评估实体类
 */
@TableName("amlewsbdl_riskassessment")
@Data
public class RiskAssessment extends BaseEntity {
    private static final long serialVersionUID = -4844874992503875797LL;
    // 评估 ID
```

```
    private Integer assessmentId;
    // 评估因素
    private String factors;
    // 风险等级
    private String riskLevel;
    // 缓解方案
    private String mitigationPlan;
    // 客户 ID
    private Integer customerId;
    // 风险评分
    private Integer riskScore;
    // 备注
    private String notes;
    // 下次评估日期
    private String nextReviewDate;
    // 评估日期
    private Date assessmentDate;
}
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/myba
tis-3-mapper.dtd">
<mapper namespace="com.chengyu.amlewsbdl.mapper.RiskEventMapper">
    <resultMap type="RiskEvent" id="RiskEventResult">
        <result property="eventId" column="event_id"    />
        <result property="notes" column="notes"    />
        <result property="impactLevel" column="impact_level"    />
        <result property="eventTime" column="event_time"    />
        <result property="customerId" column="customer_id"    />
        <result property="merchantId" column="merchant_id"    />
        <result property="details" column="details"    />
        <result property="handled" column="handled"    />
        <result property="eventType" column="event_type"    />
    </resultMap>
    <sql id="selectRiskEventVo">
        select event_id, notes, impact_level, event_time, customer_id, merchant_id, detai
ls, handled, event_type
        from amlewsbdl_riskevent
    </sql>
    <select id="selectRiskEventList" parameterType="RiskEvent" resultMap="RiskEventResult
">
        <include refid="selectRiskEventVo"/>
        <where>
            <if test="eventId != null ">
            and event_id = #{eventId}
            </if>
            <if test="notes != null and notes != ''">
            and notes = #{notes}
            </if>
            <if test="impactLevel != null and impactLevel != ''">
            and impact_level = #{impactLevel}
```

```
            </if>
            <if test="eventTime != null ">
            and event_time = #{eventTime}
            </if>
            <if test="customerId != null ">
            and customer_id = #{customerId}
            </if>
            <if test="merchantId != null ">
            and merchant_id = #{merchantId}
            </if>
            <if test="details != null and details != ''">
            and details = #{details}
            </if>
            <if test="handled != null ">
            and handled = #{handled}
            </if>
            <if test="eventType != null and eventType != ''">
            and event_type = #{eventType}
            </if>
        </where>
    </select>
    <delete id="deleteRiskEventByEventId" parameterType="Integer">
        delete from am1ewsbd1_riskevent where event_id = #{eventId}
    </delete>

    <insert id="insertRiskEvent" parameterType="RiskEvent" useGeneratedKeys="true" keyPro
perty="eventId">
        insert into am1ewsbd1_riskevent
        <trim prefix="(" suffix=")" suffixOverrides=",">
            <if test="notes != null and notes != ''">notes,</if>
            <if test="impactLevel != null and impactLevel != ''">impact_level,</if>
            <if test="eventTime != null">event_time,</if>
            <if test="customerId != null">customer_id,</if>
            <if test="merchantId != null">merchant_id,</if>
            <if test="details != null and details != ''">details,</if>
            <if test="handled != null">handled,</if>
            <if test="eventType != null and eventType != ''">event_type,</if>
         </trim>
        <trim prefix="values (" suffix=")" suffixOverrides=",">
            <if test="notes != null and notes != ''">#{notes},</if>
            <if test="impactLevel != null and impactLevel != ''">#{impactLevel},</if>
            <if test="eventTime != null">#{eventTime},</if>
            <if test="customerId != null">#{customerId},</if>
            <if test="merchantId != null">#{merchantId},</if>
            <if test="details != null and details != ''">#{details},</if>
            <if test="handled != null">#{handled},</if>
            <if test="eventType != null and eventType != ''">#{eventType},</if>
         </trim>
    </insert>
    <delete id="deleteRiskEventByEventIds" parameterType="String">
```

```
        delete from am1ewsbd1_riskevent where event_id in
        <foreach item="eventId" collection="array" open="(" separator="," close=")">
            #{eventId}
        </foreach>
    </delete>
</mapper>
package com.chengyu.am1ewsbd1.controller;
import java.util.List;
import javax.servlet.http.HttpServletResponse;
import com.chengyu.am1ewsbd1.common.core.domain.RespResult;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RestController;
import com.chengyu.am1ewsbd1.common.annotation.Log;
import com.chengyu.am1ewsbd1.common.core.controller.BaseController;
import com.chengyu.am1ewsbd1.common.utils.poi.ExcelUtil;
import com.chengyu.am1ewsbd1.common.core.page.TableDataInfo;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.GetMapping;
import com.chengyu.am1ewsbd1.common.enums.BusinessType;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import com.chengyu.am1ewsbd1.domain.Customer;
import com.chengyu.am1ewsbd1.service.ICustomerService;
import org.springframework.web.bind.annotation.PathVariable;
@RestController
@RequestMapping("/customer")
public class CustomerController extends BaseAction {
    @Autowired
    private ICustomerService customerService;

    /**
     * 查询客户信息列表
     */
    @GetMapping("/list")
    public TableDataInfo customerList(CustomerQueryVo queryVo){
        startPage();
        List<Customer> customerList = customerService.getCustomerList(queryVo);
        list.forEach(i -> {
            i.setCustomerId(CustomerConvUtil(i.getCustomerId));
            i.setAccountType(CustomerConvUtil(i.getAccountType));
            i.setTotalTransactions(CustomerConvUtil(i.getTotalTransactions));
            i.setGender(CustomerConvUtil(i.getGender));
            i.setDateOfBirth(CustomerConvUtil(i.getDateOfBirth));
        });
        PageInfo<Customer> pageInfo = new PageInfo<>(list);
        return RespResult.pageResult(list, pageInfo.getTotal());
    }
```

```
    @PostMapping
    public RespResult addCustomer(@Valid @RequestBody CustomerAddVo addVo){
       customerService.saveCustomer(addVo);
       return RespResult.success();
    }

    public RespResult listcustomer(Customer customer){
        List<Customer> list = customerService.getCustomerList(customer);
        return RespResult.success(list);
    }

    @DeleteMapping("/{customerIds}")
    public RespResult delcustomer(@PathVariable Integer[] customerIds){
        customerService.deleteCustomerByCustomerIds(customerIds);
        return RespResult.success();
    }

    @PutMapping
    public RespResult updatecustomer(@Valid @RequestBody Customer customer){
        customerService.updateCustomer(customer);
        return RespResult.success();
    }

    @PostMapping("/export")
    public void exportCustomer(HttpServletResponse response, Customer customer){
        List<Customer> list = customerService.getCustomerList(customer);
        list.forEach(i -> {
            i.setCustomerId(CustomerExportFormat(i.getCustomerId));
            i.setAccountType(CustomerExportFormat(i.getAccountType));
            i.setTotalTransactions(CustomerExportFormat(i.getTotalTransactions));
            i.setGender(CustomerExportFormat(i.getGender));
            i.setDateOfBirth(CustomerExportFormat(i.getDateOfBirth));
        });
        ExcelUtil<Customer> excelUtil = new ExcelUtil<Customer>(Customer.class);
        util.exportExcel(response, list, "客户信息数据");
    }

    @GetMapping(value = "/{customerId}")
    public RespResult getcustomerId(@PathVariable("customerId") Integer customerId){
        return RespResult.success(customerService.getCustomerByCustomerId(customerId));
    }
}
package com.chengyu.amlewsbdl.mapper;
import java.util.List;
import com.chengyu.amlewsbdl.domain.TransactionRecord;
import com.baomidou.mybatisplus.core.mapper.BaseMapper;
/**
 * 交易记录 Dao 接口
 */
```

```
public interface TransactionRecordMapper extends BaseMapper<TransactionRecord>{
    /**
     * 新增交易记录
     *
     * @param transactionRecord 交易记录
     * @return
     */
    int insertTransactionRecord(TransactionRecord transactionRecord);

    /**
     * 修改交易记录
     *
     * @param transactionRecord 交易记录
     * @return
     */
    int updateTransactionRecord(TransactionRecord transactionRecord);

    /**
     * 查询交易记录列表
     *
     * @param transactionId 交易记录主键
     * @return
     */
    TransactionRecord selectTransactionRecordByTransactionId(Integer transactionId);

    /**
     * 根据条件查询交易记录
     */
    List<TransactionRecordVo> selectByCondition(TransactionRecordQueryDto queryTransactio
nRecordDto);

    /**
     * 删除交易记录
     * @param transactionId 交易记录主键
     * @return
     */
    int deleteTransactionRecordByTransactionId(Integer transactionId);

    /**
     * 批量删除交易记录
     * @param transactionIds  ID集合
     * @return
     */
    int deleteTransactionRecordByTransactionIds(Integer[] transactionIds);
}

package com.chengyu.am1ewsbd1.flink;
import org.apache.flink.api.common.eventtime.WatermarkStrategy;
import org.apache.flink.api.common.functions.FlatMapFunction;
import org.apache.flink.api.common.restartstrategy.RestartStrategies;
```

```
import org.apache.flink.api.common.serialization.SimpleStringSchema;
import org.apache.flink.api.java.tuple.Tuple3;
import org.apache.flink.api.java.tuple.Tuple4;
import org.apache.flink.connector.kafka.source.enumerator.initializer.OffsetsInitializer;
import org.apache.flink.streaming.api.datastream.DataStream;
import org.apache.flink.streaming.api.environment.StreamExecutionEnvironment;
import org.apache.flink.streaming.api.windowing.assigners.TumblingProcessingTimeWindows;
import org.apache.flink.streaming.api.windowing.time.Time;
import org.apache.flink.streaming.connectors.kafka.FlinkKafkaConsumer;
import org.apache.flink.connector.kafka.source.KafkaSource;
import org.apache.flink.util.Collector;
import org.example.sql.DatabaseSink;
import java.sql.Timestamp;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Properties;
import java.util.concurrent.TimeUnit;
public class StreamFlinkConn {
    public static void flinkConn() throws Exception {
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm");
        SimpleDateFormat sdf_hour = new SimpleDateFormat("yyyy-MM-dd HH");
        final StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnv
ironment();
        env.setRestartStrategy(RestartStrategies.fixedDelayRestart(
                org.apache.flink.api.common.time.Time.of(10, TimeUnit.SECONDS) // 延迟
        ));
        KafkaSource<String> source = KafkaSource.<String>builder()
                .setBootstrapServers("kafka1:9092,kafka2:9092,kafka3:9092")
                .setGroupId("testGroup")
                .setTopics("new-topic")
                .setStartingOffsets(OffsetsInitializer.earliest())
                .setValueOnlyDeserializer(new SimpleStringSchema())
                .build();
        env.fromSource(source, WatermarkStrategy.noWatermarks(), "Kafka Source");
        DataStream<String> stream = env.fromSource(source, WatermarkStrategy.noWatermarks
(), "Kafka Source");
        counts.print( );
        DataStream<Tuple3<Timestamp,Long ,Double>> transactionVolumes = stream
                .flatMap(new FlatMapFunction<String, Tuple3<Timestamp,Long ,Double>>() {
                    @Override
                    public void flatMap(String line, Collector<Tuple3<Timestamp,Long ,Dou
ble>> out) {
                        try {
                            String[] fields = line.split(",");
                            String s = fields[0];
                            Date date = sdf.parse(s);
                            Timestamp sqlTimestamp = new Timestamp(date.getTime());
                            double price = Double.parseDouble(fields[3]);
                            long quantity = Long.parseLong(fields[4]);
                            double amount = price * quantity;
```

```
                            out.collect(Tuple3.of(sqlTimestamp,quantity, amount));
                        } catch (Exception e) {
                            System.out.println(line);
                        }
                    }
                });
        // 计算每 10 毫秒的数据
        DataStream<Tuple3<Timestamp,Long ,Double>> oneSecondAmounts = transactionVolumes
                .windowAll(TumblingProcessingTimeWindows.of(Time.milliseconds(500)))
                .reduce((Tuple3<Timestamp,Long ,Double> value1, Tuple3<Timestamp,Long ,Do
uble> value2) -> {
                    return Tuple3.of(value1.f0,value1.f1 + value2.f1, value1.f2 + value2.
f2);
                });
        oneSecondAmounts.print();
        DatabaseSink dbSink = new DatabaseSink("jdbc:mysql://rm-bp1it5e2bjw5qgy58no.mysql.r
ds.aliyuncs.com:3306/", "IKUN", "I+kun=ikun");
        dbSink.addSink(oneSecondAmounts);
        DataStream<Tuple4<String,String, Timestamp, Double>> stockVolumes = stream
                .flatMap(new FlatMapFunction<String, Tuple4<String,String, Timestamp, Dou
ble>>() {
                    @Override
                    public void flatMap(String line, Collector<Tuple4<String,String, Time
stamp, Double>> out) {
                        try {
                            String[] fields = line.split(",");
                            String s = fields[0];
                            Date date = sdf.parse(s);
                            Timestamp sqlTimestamp = new Timestamp(date.getTime());
                            String stock_name = fields[2];
                            String stock_code=fields[1];
                            double price = Double.parseDouble(fields[3]);
                            long quantity = Long.parseLong(fields[4]);
                            double amount = price * quantity;
                            out.collect(Tuple4.of(stock_code,stock_name, sqlTimestamp, am
ount));
                        } catch (Exception e) {
                            System.out.println(line);
                        }
                    }
                });
        DataStream<Tuple4<String,String,Timestamp, Double>> oneMinuteVolumes = stockVolum
es
                .keyBy(t -> t.f0)
                .window(TumblingProcessingTimeWindows.of(Time.milliseconds(500)))
                .reduce((Tuple4<String,String, Timestamp, Double> value1, Tuple4<String,S
tring, Timestamp, Double> value2) -> {
                    return Tuple4.of(value1.f0,value1.f1, value2.f2, value1.f3 + value2.f
3);
                })
```

```
                ;
        stockVolumes.print();
        dbSink.addmin_stock(oneMinuteVolumes);
        DataStream<Tuple3<Timestamp,String, Long>> stock_platform = stream
                .flatMap(new FlatMapFunction<String, Tuple3<Timestamp,String, Long>>() {
                    @Override
                    public void flatMap(String line, Collector<Tuple3<Timestamp,String, L
ong>> out) {
                        try {
                            String[] fields = line.split(",");
                            String s = fields[0];
                            Date date = sdf.parse(s);
                            Timestamp sqlTimestamp = new Timestamp(date.getTime());
                            String platform = fields[7];
                            long quantity = Long.parseLong(fields[4]);
                            out.collect(Tuple3.of(sqlTimestamp,platform,quantity));
                        } catch (Exception e) {
                            System.out.println(line);
                        }
                    }
                });
        DataStream<Tuple3<Timestamp,String, Long>> oneMinuteplatform = stock_platform
                .keyBy(t -> t.f1)
                .window(TumblingProcessingTimeWindows.of(Time.milliseconds(500)))
                .reduce((Tuple3<Timestamp,String, Long> value1, Tuple3<Timestamp,String,
Long> value2) -> {
                    return Tuple3.of(value1.f0,value1.f1, value1.f2 + value2.f2);
                })
                ;
        oneMinuteplatform.print();
        dbSink.addmin_platform(oneMinuteplatform);
        DataStream<Tuple3<Timestamp,String, Long>> stock_type = stream
                .flatMap(new FlatMapFunction<String, Tuple3<Timestamp,String, Long>>() {
                    @Override
                    public void flatMap(String line, Collector<Tuple3<Timestamp,String, L
ong>> out) {
                        try {
                            String[] fields = line.split(",");
                            String s = fields[0];
                            Date date = sdf_hour.parse(s);
                            Timestamp sqlTimestamp = new Timestamp(date.getTime());
                            String type = fields[8];
                            long quantity = Long.parseLong(fields[4]);
                            out.collect(Tuple3.of(sqlTimestamp,type,quantity));
                        } catch (Exception e) {
                            System.out.println(line);
                        }
                    }
                });
        DataStream<Tuple3<Timestamp,String, Long>> oneMinutetype =stock_type
```

```
                .keyBy(t -> t.f1)
                .window(TumblingProcessingTimeWindows.of(Time.milliseconds(500)))
                .reduce((Tuple3<Timestamp,String, Long> value1, Tuple3<Timestamp,String,
Long> value2) -> {
                    return Tuple3.of(value1.f0,value1.f1, value1.f2 + value2.f2);
                })
                ;
        dbSink.addday_type(oneMinutetype);
        DataStream<Tuple3<Timestamp,String, Long>> stock_place=stream
                .flatMap(new FlatMapFunction<String, Tuple3<Timestamp,String, Long>>() {
                    @Override
                    public void flatMap(String line, Collector<Tuple3<Timestamp,String, L
ong>> out) {
                        try {
                            String[] fields = line.split(",");
                            String s = fields[0];
                            Date date = sdf_hour.parse(s);
                            Timestamp sqlTimestamp = new Timestamp(date.getTime());
                            String place = fields[6];
                            if (place.length() >= 3) {
                                place = place.substring(0, place.length() - 1);
                            }
                            long quantity = Long.parseLong(fields[4]);
                            out.collect(Tuple3.of(sqlTimestamp,place,quantity));
                        } catch (Exception e) {
                            System.out.println(line);
                        }
                    }
                });
        DataStream<Tuple3<Timestamp,String, Long>> oneMinuteplace =stock_place
                .keyBy(t -> t.f1)
                .window(TumblingProcessingTimeWindows.of(Time.milliseconds(500)))
                .reduce((Tuple3<Timestamp,String, Long> value1, Tuple3<Timestamp,String,
Long> value2) -> {
                    return Tuple3.of(value1.f0,value1.f1, value1.f2 + value2.f2);
                })
                ;
        dbSink.addday_place(oneMinuteplace);
        DataStream<Tuple3<Timestamp,String, Long>> stock_transaction=stream
                .flatMap(new FlatMapFunction<String, Tuple3<Timestamp,String, Long>>() {
                    @Override
                    public void flatMap(String line, Collector<Tuple3<Timestamp,String, L
ong>> out) {
                        try {
                            String[] fields = line.split(",");
                            String s = fields[0];
                            Date date = sdf.parse(s);
                            Timestamp sqlTimestamp = new Timestamp(date.getTime());
                            String type = fields[5];
                            long quantity = Long.parseLong(fields[4]);
```

```
                            out.collect(Tuple3.of(sqlTimestamp,type,quantity));
                        } catch (Exception e) {
                            System.out.println(line);
                        }
                    }
                });
        DataStream<Tuple3<Timestamp,String, Long>> oneMinutransaction =stock_transaction
                .keyBy(t -> t.f1)
                .window(TumblingProcessingTimeWindows.of(Time.milliseconds(500)))
                .reduce((Tuple3<Timestamp,String, Long> value1, Tuple3<Timestamp,String,
Long> value2) -> {
                    return Tuple3.of(value1.f0,value1.f1, value1.f2 + value2.f2);
                });
    }
}
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/myba
tis-3-mapper.dtd">
<mapper namespace="com.chengyu.am1ewsbd1.mapper.RiskAssessmentMapper">
    <resultMap type="RiskAssessment" id="RiskAssessmentResult">
        <result property="assessmentId" column="assessment_id"    />
        <result property="factors" column="factors"    />
        <result property="riskLevel" column="risk_level"    />
        <result property="mitigationPlan" column="mitigation_plan"    />
        <result property="customerId" column="customer_id"    />
        <result property="riskScore" column="risk_score"    />
        <result property="notes" column="notes"    />
        <result property="nextReviewDate" column="next_review_date"    />
        <result property="assessmentDate" column="assessment_date"    />
    </resultMap>
    <sql id="selectRiskAssessmentVo">
        select assessment_id, factors, risk_level, mitigation_plan, customer_id, risk_sco
re, notes, next_review_date, assessment_date
        from am1ewsbd1_riskassessment
    </sql>
    <select id="selectRiskAssessmentList" parameterType="RiskAssessment" resultMap="RiskA
ssessmentResult">
        <include refid="selectRiskAssessmentVo"/>
        <where>
            <if test="assessmentId != null ">
            and assessment_id = #{assessmentId}
            </if>
            <if test="factors != null and factors != ''">
            and factors = #{factors}
            </if>
            <if test="riskLevel != null and riskLevel != ''">
            and risk_level = #{riskLevel}
            </if>
            <if test="mitigationPlan != null and mitigationPlan != ''">
            and mitigation_plan = #{mitigationPlan}
```

```
            </if>
            <if test="customerId != null ">
            and customer_id = #{customerId}
            </if>
            <if test="riskScore != null ">
            and risk_score = #{riskScore}
            </if>
            <if test="notes != null and notes != ''">
            and notes = #{notes}
            </if>
            <if test="nextReviewDate != null and nextReviewDate != ''">
            and next_review_date = #{nextReviewDate}
            </if>
            <if test="assessmentDate != null ">
            and assessment_date = #{assessmentDate}
            </if>
        </where>
    </select>
    <delete id="deleteRiskAssessmentByAssessmentId" parameterType="Integer">
        delete from am1ewsbd1_riskassessment where assessment_id = #{assessmentId}
    </delete>

    <insert id="insertRiskAssessment" parameterType="RiskAssessment" useGeneratedKeys="tr
ue" keyProperty="assessmentId">
        insert into am1ewsbd1_riskassessment
        <trim prefix="(" suffix=")" suffixOverrides=",">
            <if test="factors != null and factors != ''">factors,</if>
            <if test="riskLevel != null and riskLevel != ''">risk_level,</if>
            <if test="mitigationPlan != null and mitigationPlan != ''">mitigation_plan,</
if>
            <if test="customerId != null">customer_id,</if>
            <if test="riskScore != null">risk_score,</if>
            <if test="notes != null and notes != ''">notes,</if>
            <if test="nextReviewDate != null and nextReviewDate != ''">next_review_date,<
/if>
            <if test="assessmentDate != null">assessment_date,</if>
         </trim>
        <trim prefix="values (" suffix=")" suffixOverrides=",">
            <if test="factors != null and factors != ''">#{factors},</if>
            <if test="riskLevel != null and riskLevel != ''">#{riskLevel},</if>
            <if test="mitigationPlan != null and mitigationPlan != ''">#{mitigationPlan},
</if>
            <if test="customerId != null">#{customerId},</if>
            <if test="riskScore != null">#{riskScore},</if>
            <if test="notes != null and notes != ''">#{notes},</if>
            <if test="nextReviewDate != null and nextReviewDate != ''">#{nextReviewDate},
</if>
            <if test="assessmentDate != null">#{assessmentDate},</if>
         </trim>
    </insert>
```

```
    <delete id="deleteRiskAssessmentByAssessmentIds" parameterType="String">
        delete from am1ewsbd1_riskassessment where assessment_id in
        <foreach item="assessmentId" collection="array" open="(" separator="," close=")">
            #{assessmentId}
        </foreach>
    </delete>
</mapper>
package com.chengyu.am1ewsbd1.controller;
import java.util.List;
import javax.servlet.http.HttpServletResponse;
import com.chengyu.am1ewsbd1.common.core.domain.RespResult;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RestController;
import com.chengyu.am1ewsbd1.common.annotation.Log;
import com.chengyu.am1ewsbd1.common.core.controller.BaseController;
import com.chengyu.am1ewsbd1.common.utils.poi.ExcelUtil;
import com.chengyu.am1ewsbd1.common.core.page.TableDataInfo;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.GetMapping;
import com.chengyu.am1ewsbd1.common.enums.BusinessType;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import com.chengyu.am1ewsbd1.domain.Merchant;
import com.chengyu.am1ewsbd1.service.IMerchantService;
import org.springframework.web.bind.annotation.PathVariable;
@RestController
@RequestMapping("/merchant")
public class MerchantController extends BaseAction {
    @Autowired
    private IMerchantService merchantService;

    /**
     * 查询商户信息列表
     */
    @GetMapping("/list")
    public TableDataInfo merchantList(MerchantQueryVo queryVo){
        startPage();
        List<Merchant> merchantList = merchantService.getMerchantList(queryVo);
        list.forEach(i -> {
            i.setComplianceOfficer(MerchantConvUtil(i.getComplianceOfficer));
            i.setRiskRating(MerchantConvUtil(i.getRiskRating));
            i.setNumberOfTransactions(MerchantConvUtil(i.getNumberOfTransactions));
            i.setPhoneNumber(MerchantConvUtil(i.getPhoneNumber));
            i.setEstablishmentDate(MerchantConvUtil(i.getEstablishmentDate));
            i.setBusinessType(MerchantConvUtil(i.getBusinessType));
            i.setAverageMonthlyTransactions(MerchantConvUtil(i.getAverageMonthlyTransacti
ons));
```

```
            i.setName(MerchantConvUti1(i.getName));
            i.setRegistrationNumber(MerchantConvUti1(i.getRegistrationNumber));
        });
        PageInfo<Merchant> pageInfo = new PageInfo<>(list);
        return RespResult.pageResult(list, pageInfo.getTotal());
    }

    @PostMapping
    public RespResult addMerchant(@Valid @RequestBody MerchantAddVo addVo){
       merchantService.saveMerchant(addVo);
       return RespResult.success();
    }

    public RespResult listmerchant(Merchant merchant){
        List<Merchant> list = merchantService.getMerchantList(merchant);
        return RespResult.success(list);
    }

    @DeleteMapping("/{merchantIds}")
    public RespResult delmerchant(@PathVariable Integer[] merchantIds){
        merchantService.deleteMerchantByMerchantIds(merchantIds);
        return RespResult.success();
    }

    @PutMapping
    public RespResult updatemerchant(@Valid @RequestBody Merchant merchant){
        merchantService.updateMerchant(merchant);
        return RespResult.success();
    }

    @PostMapping("/export")
    public void exportMerchant(HttpServletResponse response, Merchant merchant){
        List<Merchant> list = merchantService.getMerchantList(merchant);
        list.forEach(i -> {
            i.setComplianceOfficer(MerchantExportFormat(i.getComplianceOfficer));
            i.setRiskRating(MerchantExportFormat(i.getRiskRating));
            i.setNumberOfTransactions(MerchantExportFormat(i.getNumberOfTransactions));
            i.setPhoneNumber(MerchantExportFormat(i.getPhoneNumber));
            i.setEstablishmentDate(MerchantExportFormat(i.getEstablishmentDate));
            i.setBusinessType(MerchantExportFormat(i.getBusinessType));
            i.setAverageMonthlyTransactions(MerchantExportFormat(i.getAverageMonthlyTrans
actions));
            i.setName(MerchantExportFormat(i.getName));
            i.setRegistrationNumber(MerchantExportFormat(i.getRegistrationNumber));
        });
        ExcelUtil<Merchant> excelUtil = new ExcelUtil<Merchant>(Merchant.class);
        util.exportExcel(response, list, "商户信息数据");
    }

    @GetMapping(value = "/{merchantId}")
```

```
    public RespResult getmerchantId(@PathVariable("merchantId") Integer merchantId){
        return RespResult.success(merchantService.getMerchantByMerchantId(merchantId));
    }
}
package com.chengyu.amlewsbdl.mapper;
import java.util.List;
import com.chengyu.amlewsbdl.domain.RiskAssessment;
import com.baomidou.mybatisplus.core.mapper.BaseMapper;
/**
 * 风险评估 Dao 接口
 */
public interface RiskAssessmentMapper extends BaseMapper<RiskAssessment>{
    /**
     * 新增风险评估
     *
     * @param riskAssessment 风险评估
     * @return
     */
    int insertRiskAssessment(RiskAssessment riskAssessment);

    /**
     * 修改风险评估
     *
     * @param riskAssessment 风险评估
     * @return
     */
    int updateRiskAssessment(RiskAssessment riskAssessment);

    /**
     * 查询风险评估列表
     *
     * @param assessmentId 风险评估主键
     * @return
     */
    RiskAssessment selectRiskAssessmentByAssessmentId(Integer assessmentId);

    /**
     * 根据条件查询风险评估
     */
    List<RiskAssessmentVo> selectByCondition(RiskAssessmentQueryDto queryRiskAssessmentDt
o);

    /**
     * 删除风险评估
     * @param assessmentId 风险评估主键
     * @return
     */
    int deleteRiskAssessmentByAssessmentId(Integer assessmentId);

    /**
```

```
     * 批量删除风险评估
     * @param assessmentIds  ID 集合
     * @return
     */
    int deleteRiskAssessmentByAssessmentIds(Integer[] assessmentIds);
}

package com.chengyu.amlewsbdl.mapper;
import java.util.List;
import com.chengyu.amlewsbdl.domain.TransactionFeature;
import com.baomidou.mybatisplus.core.mapper.BaseMapper;
/**
 * 交易特征 Dao 接口
 */
public interface TransactionFeatureMapper extends BaseMapper<TransactionFeature>{
    /**
     * 新增交易特征
     *
     * @param transactionFeature 交易特征
     * @return
     */
    int insertTransactionFeature(TransactionFeature transactionFeature);

    /**
     * 修改交易特征
     *
     * @param transactionFeature 交易特征
     * @return
     */
    int updateTransactionFeature(TransactionFeature transactionFeature);

    /**
     * 查询交易特征列表
     *
     * @param featureId 交易特征主键
     * @return
     */
    TransactionFeature selectTransactionFeatureByFeatureId(Integer featureId);

    /**
     * 根据条件查询交易特征
     */
    List<TransactionFeatureVo> selectByCondition(TransactionFeatureQueryDto queryTransact
ionFeatureDto);

    /**
     * 删除交易特征
     * @param featureId 交易特征主键
     * @return
     */
```

```
    int deleteTransactionFeatureByFeatureId(Integer featureId);

    /**
     * 批量删除交易特征
     * @param featureIds  ID 集合
     * @return
     */
    int deleteTransactionFeatureByFeatureIds(Integer[] featureIds);
}

<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.chengyu.amlewsbdl.mapper.MerchantMapper">
    <resultMap type="Merchant" id="MerchantResult">
        <result property="merchantId" column="merchant_id"    />
        <result property="businessAddress" column="business_address"    />
        <result property="email" column="email"    />
        <result property="numberOfTransactions" column="number_of_transactions"    />
        <result property="establishmentDate" column="establishment_date"    />
        <result property="website" column="website"    />
        <result property="industryCategory" column="industry_category"    />
        <result property="averageMonthlyTransactions" column="average_monthly_transactions"    />
        <result property="complianceOfficer" column="compliance_officer"    />
        <result property="phoneNumber" column="phone_number"    />
        <result property="businessType" column="business_type"    />
        <result property="riskRating" column="risk_rating"    />
        <result property="registrationNumber" column="registration_number"    />
        <result property="name" column="name"    />
    </resultMap>
    <sql id="selectMerchantVo">
        select merchant_id, business_address, email, number_of_transactions, establishment_date, website, industry_category, average_monthly_transactions, compliance_officer, phone_number, business_type, risk_rating, registration_number, name
        from amlewsbdl_merchant
    </sql>
    <select id="selectMerchantList" parameterType="Merchant" resultMap="MerchantResult">
        <include refid="selectMerchantVo"/>
        <where>
            <if test="merchantId != null ">
            and merchant_id = #{merchantId}
            </if>
            <if test="businessAddress != null and businessAddress != ''">
            and business_address = #{businessAddress}
            </if>
            <if test="email != null and email != ''">
            and email = #{email}
            </if>
            <if test="numberOfTransactions != null ">
```

```
            and number_of_transactions = #{numberOfTransactions}
            </if>
            <if test="establishmentDate != null and establishmentDate != ''">
            and establishment_date = #{establishmentDate}
            </if>
            <if test="website != null and website != ''">
            and website = #{website}
            </if>
            <if test="industryCategory != null and industryCategory != ''">
            and industry_category = #{industryCategory}
            </if>
            <if test="averageMonthlyTransactions != null ">
            and average_monthly_transactions = #{averageMonthlyTransactions}
            </if>
            <if test="complianceOfficer != null and complianceOfficer != ''">
            and compliance_officer = #{complianceOfficer}
            </if>
            <if test="phoneNumber != null and phoneNumber != ''">
            and phone_number = #{phoneNumber}
            </if>
            <if test="businessType != null and businessType != ''">
            and business_type = #{businessType}
            </if>
            <if test="riskRating != null ">
            and risk_rating = #{riskRating}
            </if>
            <if test="registrationNumber != null and registrationNumber != ''">
            and registration_number = #{registrationNumber}
            </if>
            <if test="name != null and name != ''">
            and name = #{name}
            </if>
        </where>
    </select>
    <delete id="deleteMerchantByMerchantId" parameterType="Integer">
        delete from am1ewsbd1_merchant where merchant_id = #{merchantId}
    </delete>

    <insert id="insertMerchant" parameterType="Merchant" useGeneratedKeys="true" keyPrope
rty="merchantId">
        insert into am1ewsbd1_merchant
        <trim prefix="(" suffix=")" suffixOverrides=",">
            <if test="businessAddress != null and businessAddress != ''">business_address,<
/if>
            <if test="email != null and email != ''">email,</if>
            <if test="numberOfTransactions != null">number_of_transactions,</if>
            <if test="establishmentDate != null and establishmentDate != ''">establishmen
t_date,</if>
            <if test="website != null and website != ''">website,</if>
            <if test="industryCategory != null and industryCategory != ''">industry_categ
```

```
ory,</if>
            <if test="averageMonthlyTransactions != null">average_monthly_transactions,</
if>
            <if test="complianceOfficer != null and complianceOfficer != ''">compliance_o
fficer,</if>
            <if test="phoneNumber != null and phoneNumber != ''">phone_number,</if>
            <if test="businessType != null and businessType != ''">business_type,</if>
            <if test="riskRating != null">risk_rating,</if>
            <if test="registrationNumber != null and registrationNumber != ''">registrati
on_number,</if>
            <if test="name != null and name != ''">name,</if>
         </trim>
        <trim prefix="values (" suffix=")" suffixOverrides=",">
            <if test="businessAddress != null and businessAddress != ''">#{businessAddres
s},</if>
            <if test="email != null and email != ''">#{email},</if>
            <if test="numberOfTransactions != null">#{numberOfTransactions},</if>
            <if test="establishmentDate != null and establishmentDate != ''">#{establishm
entDate},</if>
            <if test="website != null and website != ''">#{website},</if>
            <if test="industryCategory != null and industryCategory != ''">#{industryCate
gory},</if>
            <if test="averageMonthlyTransactions != null">#{averageMonthlyTransactions},<
/if>
            <if test="complianceOfficer != null and complianceOfficer != ''">#{compliance
Officer},</if>
            <if test="phoneNumber != null and phoneNumber != ''">#{phoneNumber},</if>
            <if test="businessType != null and businessType != ''">#{businessType},</if>
            <if test="riskRating != null">#{riskRating},</if>
            <if test="registrationNumber != null and registrationNumber != ''">#{registra
tionNumber},</if>
            <if test="name != null and name != ''">#{name},</if>
         </trim>
    </insert>
    <delete id="deleteMerchantByMerchantIds" parameterType="String">
        delete from am1ewsbd1_merchant where merchant_id in
        <foreach item="merchantId" collection="array" open="(" separator="," close=")">
            #{merchantId}
        </foreach>
    </delete>
</mapper>
package com.chengyu.am1ewsbd1.mapper;
import java.util.List;
import com.chengyu.am1ewsbd1.domain.Merchant;
import com.baomidou.mybatisplus.core.mapper.BaseMapper;
/**
 * 商户信息 Dao 接口
 */
public interface MerchantMapper extends BaseMapper<Merchant>{
    /**
```

```
     * 新增商户信息
     *
     * @param merchant 商户信息
     * @return
     */
    int insertMerchant(Merchant merchant);

    /**
     * 修改商户信息
     *
     * @param merchant 商户信息
     * @return
     */
    int updateMerchant(Merchant merchant);

    /**
     * 查询商户信息列表
     *
     * @param merchantId 商户信息主键
     * @return
     */
    Merchant selectMerchantByMerchantId(Integer merchantId);

    /**
     * 根据条件查询商户信息
     */
    List<MerchantVo> selectByCondition(MerchantQueryDto queryMerchantDto);

    /**
     * 删除商户信息
     * @param merchantId 商户信息主键
     * @return
     */
    int deleteMerchantByMerchantId(Integer merchantId);

    /**
     * 批量删除商户信息
     * @param merchantIds  ID集合
     * @return
     */
    int deleteMerchantByMerchantIds(Integer[] merchantIds);
}

package com.chengyu.am1ewsbd1.service.impl;
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;
import com.chengyu.am1ewsbd1.mapper.TransactionRecordMapper;
import com.chengyu.am1ewsbd1.domain.TransactionRecord;
import com.chengyu.am1ewsbd1.service.ITransactionRecordService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
```

```
import java.util.List;
@Service
public class TransactionRecordServiceImpl extends ServiceImpl<TransactionRecordMapper, Tr
ansactionRecord> implements ITransactionRecordService{
    @Autowired
    private TransactionRecordMapper transactionRecordMapper;

    @Override
    public TransactionRecord getTransactionRecordByTransactionId(Integer transactionId){
        return transactionRecordMapper.selectTransactionRecordByTransactionId(transaction
Id);
    }

    @Override
    public List<TransactionRecord> listTransactionRecord(TransactionRecord transactionRec
ord){
        return transactionRecordMapper.selectTransactionRecordList(transactionRecord);
    }

    @Transactional
    @Override
    public int saveTransactionRecord(TransactionRecord transactionRecord){
        return transactionRecordMapper.insertTransactionRecord(transactionRecord);
    }

    private void checkExisted(TransactionRecordEntity entity) {
        TransactionRecordEntity existEntity = getOne(
                Wrappers.lambdaQuery(TransactionRecordEntity.class)
                .eq(TransactionRecordEntity::getTransactionId, entity.getTransactionId)
                false);
        if (existEntity != null && !existEntity.getId().equals(entity.getId())) {
            throw new BizException("交易记录记录已存在！");
        }
    }

    @Override
    @Transactional
    public int delTransactionRecordByTransactionIds(Integer[] transactionIds){
        return transactionRecordMapper.deleteTransactionRecordByTransactionIds(transactio
nIds);
    }

    private void transactionRecordData(List<TransactionRecordDto> list) {
        List<String> list = list.stream().map(TransactionRecordDto::getTransactionId)
                .collect(Collectors.toList());
        list.stream().forEach(transactionRecord -> {
            TransactionRecord transactionRecordEntry = entryTransactionRecordMap.get(tran
sactionRecord.getTransactionId);
            if (transactionRecordEntry != null) {
                transactionRecord.setTransactionId(TransactionRecordTrans(transactionReco
```

```
rdEntry.getTransactionId));
                transactionRecord.setIpLocation(TransactionRecordTrans(transactionRecordE
ntry.getIpLocation));
                transactionRecord.setDestinationAccount(TransactionRecordTrans(transactio
nRecordEntry.getDestinationAccount));
            }
        });
    }

    @Override
    @Transactional
    public int updateTransactionRecord(TransactionRecord transactionRecord){
        return transactionRecordMapper.updateTransactionRecord(transactionRecord);
    }

    @Override
    @Transactional
    public int delTransactionRecordByTransactionId(Integer transactionId){
        return transactionRecordMapper.deleteTransactionRecordByTransactionId(transaction
Id);
    }
}
package com.chengyu.amlewsbdl.util;
import java.io.File;
import java.net.URI;
import java.util.HashMap;
import java.util.Map;
import java.util.Properties;
import org.springframework.beans.BeansException;
import org.springframework.beans.factory.config.ConfigurableListableBeanFactory;
import org.springframework.beans.factory.config.PropertyPlaceholderConfigurer;
import org.springframework.context.ApplicationContext;
import org.springframework.context.ApplicationContextAware;
/**
 * 资源工具类
 */
public class ResourceUtil extends PropertyPlaceholderConfigurer implements ApplicationCon
textAware {

    final static String TAG = "ResourceUtil";
    public static final String[] PICTURE_SUFFIX = { "png", "jpg", "jpeg" };
    private static Map<String, Object> propsMap;
    private static ApplicationContext applicationContext;

    @Override
    protected void processProperties(ConfigurableListableBeanFactory beanFactory, Propert
ies props) throws BeansException {
        super.processProperties(beanFactory, props);
        propsMap = new HashMap<String, Object>();
        for (Map.Entry<Object, Object> entry : props.entrySet()) {
```

```
            String key = entry.getKey().toString();
            Object obj = entry.getValue();
            propsMap.put(key, obj);
        }
    }

    public static Object getContextProps(String name) {
        try {
            return propsMap.get(name);
        } catch (NullPointerException e) {
            return null;
        }
    }

    public static File getPictureFile() {
        URI path = URI.create((String) getContextProps("file.path"));
        File file1 = new File(path);
        if (!file1.exists()) {
            file1.mkdirs();
        }
        return file1;
    }

    public static <T> T getBean(Class<T> classType) {
        return applicationContext.getBean(classType);
    }

    @Override
    public void setApplicationContext(ApplicationContext applicationContext) throws Beans
Exception {
        this.applicationContext = applicationContext;
    }
}
package com.chengyu.am1ewsbd1.util;
public class ThreadLoc1Cache {
    private static final ThreadLocal<Object> store;
    static {
        store = new ThreadLocal<Object>() {
            @Override
            protected Object initialValue() {
                return null;
            }
        };
    }

    public static Object get() {
        return store.get();
    }

    public static void set(Object key) {
```

```
        store.set(key);
    }
}
package com.chengyu.am1ewsbd1.common.util;
import java.io.BufferedReader;
import java.io.ByteArrayOutputStream;
import java.io.Closeable;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.Reader;
import java.io.Writer;
import java.nio.charset.StandardCharsets;
public class IOUtil {
    private static final int DEFAULT_BUFFER_SIZE = 2048;

    public static IOException close(InputStream input) {
        return close((Closeable) input);
    }

    public static IOException close(OutputStream output) {
        return close((Closeable) output);
    }

    public static IOException close(final Reader input) {
        return close((Closeable) input);
    }

    public static IOException close(final Writer output) {
        return close((Closeable) output);
    }

    public static IOException close(final Closeable closeable) {
        try {
            if (closeable != null) {
                closeable.close();
            }
        } catch (final IOException ioe) {
            return ioe;
        }
        return null;
    }

    public static byte[] toByteArray(InputStream input) throws IOException {
        ByteArrayOutputStream output = new ByteArrayOutputStream();
        copy(input, output);
        return output.toByteArray();
    }
```

```
    public static int copy(InputStream input, OutputStream output) throws IOException {
        long count = copyLarge(input, output);
        if (count > Integer.MAX_VALUE) {
            return -1;
        }
        return (int) count;
    }

    public static long copyLarge(InputStream input, OutputStream output)
            throws IOException {
        return copyLarge(input, output, new byte[DEFAULT_BUFFER_SIZE]);
    }

    public static long copyLarge(InputStream input, OutputStream output, byte[] buffer)
            throws IOException {
        long count = 0;
        int n = 0;
        while (-1 != (n = input.read(buffer))) {
            output.write(buffer, 0, n);
            count += n;
        }
        return count;
    }

    public static String toString(InputStream input) throws IOException{
        return toString(input, "UTF-8");
    }

    public static String toString(InputStream input, String encoding) throws IOException
{
        BufferedReader br = null;
        try {
            StringBuilder sb = new StringBuilder();
            br = new BufferedReader(new InputStreamReader(input, encoding));
            String line;
            while ((line = br.readLine()) != null) {
                sb.append(line).append("\n");
            }
            return sb.toString();
        } finally {
            if (br != null) {
                try {
                    br.close();
                } catch (IOException ignored) {
                }
            }
        }
    }
}
package com.chengyu.amlewsbd1.auth.config;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.oauth2.provider.token.AuthorizationServerTokenService
s;
import org.springframework.security.oauth2.provider.token.DefaultTokenServices;
import org.springframework.security.oauth2.provider.token.TokenEnhancerChain;
import org.springframework.security.oauth2.provider.token.TokenStore;
import org.springframework.security.oauth2.provider.token.store.InMemoryTokenStore;
import org.springframework.security.oauth2.provider.token.store.JwtAccessTokenConverter;
import org.springframework.security.oauth2.provider.token.store.JwtTokenStore;
import java.util.Arrays;
@Configuration
public class TokenConfig {
    private String SIGNING_KEY = "mq123";

    @Autowired
    TokenStore tokenStore;

    @Autowired
    private JwtAccessTokenConverter accessTokenConverter;

    @Bean
    public TokenStore tokenStore() {
        return new JwtTokenStore(accessTokenConverter());
    }

    @Bean
    public JwtAccessTokenConverter accessTokenConverter() {
        JwtAccessTokenConverter converter = new JwtAccessTokenConverter();
        converter.setSigningKey(SIGNING_KEY);
        return converter;
    }

    //令牌管理服务
    @Bean(name="authorizationServerTokenServicesCustom")
    public AuthorizationServerTokenServices tokenService() {
        DefaultTokenServices service=new DefaultTokenServices();
        service.setSupportRefreshToken(true);
        service.setTokenStore(tokenStore);

        TokenEnhancerChain tokenEnhancerChain = new TokenEnhancerChain();
        tokenEnhancerChain.setTokenEnhancers(Arrays.asList(accessTokenConverter));
        service.setTokenEnhancer(tokenEnhancerChain);

        service.setAccessTokenValiditySeconds(7200);
        service.setRefreshTokenValiditySeconds(259200);
        return service;
    }
```

```
}
package com.chengyu.amlewsbdl.auth.config;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.http.HttpMethod;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.config.annotation.authentication.builders.Authenticat
ionManagerBuilder;
import org.springframework.security.config.annotation.method.configuration.EnableGlobalMe
thodSecurity;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.builders.WebSecurity;
import  org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfig
urerAdapter;
import org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.NoOpPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.provisioning.InMemoryUserDetailsManager;
/**
 * @description 安全管理配置
 */
@EnableWebSecurity
@EnableGlobalMethodSecurity(securedEnabled = true,prePostEnabled = true)
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {
    @Bean
    public PasswordEncoder passwordEncoder() {
        return NoOpPasswordEncoder.getInstance();
        return new BCryptPasswordEncoder();
    }

    @Autowired
    DaoAuthenticationProviderCustom daoAuthenticationProviderCustom;


    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        auth.authenticationProvider(daoAuthenticationProviderCustom);
    }

    //配置安全拦截机制
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
                .authorizeRequests()
                .antMatchers("/r/**").authenticated()
                .antMatchers(HttpMethod.POST,"/register").permitAll()
```

```
                .anyRequest().permitAll()
                .and()
                .formLogin().successForwardUrl("/login-success");
    }
    @Bean
    @Override
    public AuthenticationManager authenticationManagerBean() throws Exception {
        return super.authenticationManagerBean();
    }
}
package com.chengyu.amlewsbdl.auth.config;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.HttpMethod;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.oauth2.config.annotation.configurers.ClientDetailsSer
viceConfigurer;
import org.springframework.security.oauth2.config.annotation.web.configuration.Authorizat
ionServerConfigurerAdapter;
import org.springframework.security.oauth2.config.annotation.web.configuration.EnableAuth
orizationServer;
import org.springframework.security.oauth2.config.annotation.web.configurers.Authorizatio
nServerEndpointsConfigurer;
import org.springframework.security.oauth2.config.annotation.web.configurers.Authorizatio
nServerSecurityConfigurer;
import org.springframework.security.oauth2.provider.token.AuthorizationServerTokenService
s;
import org.springframework.security.oauth2.provider.token.DefaultTokenServices;
import javax.annotation.Resource;
@Configuration
@EnableAuthorizationServer
public class AuthorizationServer extends AuthorizationServerConfigurerAdapter {

    @Resource(name = "authorizationServerTokenServicesCustom")
    private AuthorizationServerTokenServices authorizationServerTokenServices;

    @Autowired
    private AuthenticationManager authenticationManager;

    //客户端详情服务
    @Override
    public void configure(ClientDetailsServiceConfigurer clients) throws Exception {
        clients.inMemory()
                .withClient("XcWebApp")
                .secret("secret")
                .secret(new BCryptPasswordEncoder().encode("XcWebApp"))
                .resourceIds("xc-plus")
                .authorizedGrantTypes("authorization_code", "password", "client_credentia
```

```
ls", "implicit", "refresh_token")
                .scopes("all")
                .autoApprove(false);
    }
    //访问配置
    @Override
    public void configure(AuthorizationServerEndpointsConfigurer endpoints) {
        endpoints
                .authenticationManager(authenticationManager)
                .tokenServices(authorizationServerTokenServices)
                .allowedTokenEndpointRequestMethods(HttpMethod.POST);
    }

    //安全配置
    @Override
    public void configure(AuthorizationServerSecurityConfigurer security) {
        security
                .tokenKeyAccess("permitAll()")
                .checkTokenAccess("permitAll()")
                .allowFormAuthenticationForClients();
    }
}
package com.chengyu.amlewsbdl.domain;
import java.math.BigDecimal;
import java.util.Date;
import com.fasterxml.jackson.annotation.JsonFormat;
import lombok.Data;
import com.chengyu.amlewsbdl.common.annotation.Excel;
import com.baomidou.mybatisplus.annotation.TableName;
import com.chengyu.amlewsbdl.common.core.domain.BaseEntity;
/**
 * 交易记录实体类
 */
@TableName("amlewsbdl_transactionrecord")
@Data
public class TransactionRecord extends BaseEntity {
    private static final long serialVersionUID = -5182604700818553370LL;
    // 交易 ID
    private Integer transactionId;
    // 商户 ID
    private Integer merchantId;
    // 目标账户
    private String destinationAccount;
    // 用户代理
    private String userAgent;
    // 交易类型
    private String transactionType;
    // 货币代码
    private String currencyCode;
    // 使用的设备
```

```
    private String deviceUsed;
    // 客户 ID
    private Integer customerId;
    // 交易时间
    private Date transactionTime;
    // 交易地点
    private String location;
    // IP 地址
    private String ipLocation;
    // 交易金额
    private BigDecimal amount;
    // 来源账户
    private String sourceAccount;
    // 交易状态
    private String transactionStatus;
    // 备注
    private String notes;
}
package com.chengyu.am1ewsbd1.domain;
import lombok.Data;
import com.chengyu.am1ewsbd1.common.annotation.Excel;
import com.baomidou.mybatisplus.annotation.TableName;
import com.chengyu.am1ewsbd1.common.core.domain.BaseEntity;
/**
 * 交易特征实体类
 */
@TableName("am1ewsbd1_transactionfeature")
@Data
public class TransactionFeature extends BaseEntity {
    private static final long serialVersionUID = -1361232156315711553LL;
    // 特征 ID
    private Integer featureId;
    // 特征值
    private String value;
    // 重要性得分
    private String importance;
    // 时间窗口
    private String timeWindow;
    // 频率
    private String frequency;
    // 相关性得分
    private String correlation;
    // 特征类型
    private String featureType;
    // 异常评分
    private String anomalyScore;
    // 交易 ID
    private Integer transactionId;
    // 模型版本
    private String modelVersion;
```

```
}
package com.chengyu.aml.ewsbd1.service.impl;
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;
import com.chengyu.aml.ewsbd1.mapper.RiskEventMapper;
import com.chengyu.aml.ewsbd1.domain.RiskEvent;
import com.chengyu.aml.ewsbd1.service.IRiskEventService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;
@Service
public class RiskEventServiceImpl extends ServiceImpl<RiskEventMapper, RiskEvent> impleme
nts IRiskEventService{
    @Autowired
    private RiskEventMapper riskEventMapper;

    @Override
    public RiskEvent getRiskEventByEventId(Integer eventId){
        return riskEventMapper.selectRiskEventByEventId(eventId);
    }

    @Override
    public List<RiskEvent> listRiskEvent(RiskEvent riskEvent){
        return riskEventMapper.selectRiskEventList(riskEvent);
    }

    @Transactional
    @Override
    public int saveRiskEvent(RiskEvent riskEvent){
        return riskEventMapper.insertRiskEvent(riskEvent);
    }

    private void checkExisted(RiskEventEntity entity) {
        RiskEventEntity existEntity = getOne(
                Wrappers.lambdaQuery(RiskEventEntity.class)
                .eq(RiskEventEntity::getEventId, entity.getEventId)
                false);
        if (existEntity != null && !existEntity.getId().equals(entity.getId())) {
            throw new BizException("风险事件记录已存在！");
        }
    }

    @Override
    @Transactional
    public int delRiskEventByEventIds(Integer[] eventIds){
        return riskEventMapper.deleteRiskEventByEventIds(eventIds);
    }

    private void riskEventData(List<RiskEventDto> list) {
        List<String> list = list.stream().map(RiskEventDto::getEventId)
                .collect(Collectors.toList());
```

```
        list.stream().forEach(riskEvent -> {
            RiskEvent riskEventEntry = entryRiskEventMap.get(riskEvent.getEventId);
            if (riskEventEntry != null) {
                riskEvent.setNotes(RiskEventTrans(riskEventEntry.getNotes));
                riskEvent.setEventType(RiskEventTrans(riskEventEntry.getEventType));
            }
        });
    }

    @Override
    @Transactional
    public int updateRiskEvent(RiskEvent riskEvent){
        return riskEventMapper.updateRiskEvent(riskEvent);
    }

    @Override
    @Transactional
    public int delRiskEventByEventId(Integer eventId){
        return riskEventMapper.deleteRiskEventByEventId(eventId);
    }
}
package com.chengyu.amlewsbdl.model;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
@Entity(name = "t_tmp_users")
public class User {
    public User() {
        super();
    }
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "id")
    private Integer id;

    @Column(name = "user_name", length = 15)
    private String userName;

    @Column(name = "password", length = 20, updatable = false)
    private String password;

    @Column(name = "age")
    private Integer age;

    @Column(name = "nice_name", length = 15)
    private String niceName;

    @Column(name = "mobile", length = 11)
```

```
    private String mobile;

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getUserName() {
        return userName;
    }

    public void setUserName(String user_name) {
        this.userName = user_name;
    }

    public Integer getAge() {
        return age;
    }

    public void setAge(Integer age) {
        this.age = age;
    }

    public String getNiceName() {
        return niceName;
    }

    public void setNiceName(String nice_name) {
        this.niceName = nice_name;
    }

    public String getMobile() {
        return mobile;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public void setMobile(String mobile) {
        this.mobile = mobile;
    }
```

```
    public User(Integer id, Integer age, String mobile, String userName, String niceName)
{
        super();
        this.id = id;
        this.userName = userName;
        this.age = age;
        this.niceName = niceName;
        this.mobile = mobile;
    }
}
package com.chengyu.amlewsbdl.common.util;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.Reader;
import java.math.BigDecimal;
import java.net.HttpURLConnection;
import java.net.URL;
import java.nio.charset.Charset;
import java.nio.file.Files;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.Map;
import java.util.Properties;
import java.util.UUID;
import java.util.function.Function;
import java.util.jar.JarEntry;
import java.util.jar.JarFile;
import java.util.zip.ZipEntry;
import java.util.zip.ZipOutputStream;
public class FileUtil {
    private static final int SAFE_BYTE_LENGTH = 2048;
    private static final String TEMP_PATH = ".jvmm";

    public static String getTempPath() {
        return TEMP_PATH;
    }
    public static File createTempFile(String filename) throws IOException {
        File file = new File(TEMP_PATH, filename);
        if (!file.exists()) {
            if (file.getParentFile() != null) {
                file.getParentFile().mkdirs();
            }
            if (!file.createNewFile()) {
                throw new IOException("Can not create temp file");
```

```
            }
        }
        return file;
    }

    public static File createTempDir(String dirname) throws IOException {
        File dir = new File(TEMP_PATH, dirname);
        if (!dir.exists()) {
            if (!dir.mkdirs()) {
                throw new IOException("Can not create temp file");
            }
        }
        return dir;
    }

    public static void writeByteArrayToFile(File file, byte[] data) throws IOException {
        writeByteArrayToFile(file, data, false);
    }

    public static void writeByteArrayToFile(File file, byte[] data, boolean append) throw
s IOException {
        try (OutputStream out = openOutputStream(file, append)) {
            out.write(data);
        }
    }

    public static FileOutputStream openOutputStream(File file, boolean append) throws IOE
xception {
        if (file.exists()) {
            if (file.isDirectory()) {
                throw new IOException("File '" + file + "' exists but is a directory");
            }
            if (!file.canWrite()) {
                throw new IOException("File '" + file + "' cannot be written to");
            }
        } else {
            File parent = file.getParentFile();
            if (parent != null) {
                if (!parent.mkdirs() && !parent.isDirectory()) {
                    throw new IOException("Directory '" + parent + "' could not be create
d");
                }
            }
        }
        return new FileOutputStream(file, append);
    }

    public static String readFileToString(File file, Charset encoding) throws IOException
{
        try (FileInputStream stream = new FileInputStream(file)) {
```

```
            Reader reader = new BufferedReader(new InputStreamReader(stream, encoding));
            StringBuilder builder = new StringBuilder();
            char[] buffer = new char[8192];
            int read;
            while ((read = reader.read(buffer, 0, buffer.length)) > 0) {
                builder.append(buffer, 0, read);
            }
            return builder.toString();
        }
    }

    public static Map<String, String> readProperties(String file) throws IOException {
        return readProperties(file, null);
    }

    public static Map<String, String> readProperties(String file, String globalPrefix) th
rows IOException {
        Properties properties = new Properties();
        try (FileInputStream in = new FileInputStream(file)) {
            properties.load(in);
            Map<String, String> map = new HashMap<>(properties.size());
            properties.forEach((k, v) -> {
                String key;
                if (globalPrefix != null) {
                    if (!k.toString().startsWith(globalPrefix)) {
                        return;
                    }
                    key = k.toString().replaceFirst(globalPrefix, "");
                } else {
                    key = k.toString();
                }
                map.put(key, v.toString());
            });
            return map;
        }
    }
}
package com.com.chengyu.amlewsbdl.common.utils.html;
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.concurrent.ConcurrentHashMap;
import java.util.concurrent.ConcurrentMap;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
/**
 * HTML 过滤器，用于去除 XSS 漏洞隐患。
 */
```

```
public final class HTMLFilter {
    private static final int REGEX_FLAGS_SI = Pattern.CASE_INSENSITIVE | Pattern.DOTALL;
    private static final Pattern P_COMMENTS = Pattern.compile("<!--(.*?)-->", Pattern.DOT
ALL);
    private static final Pattern P_COMMENT = Pattern.compile("^!--(.*)--$", REGEX_FLAGS_S
I);
    private static final Pattern P_TAGS = Pattern.compile("<(.*?)>", Pattern.DOTALL);
    private static final Pattern P_END_TAG = Pattern.compile("^/([a-z0-9]+)", REGEX_FLAGS
_SI);
    private static final Pattern P_START_TAG = Pattern.compile("^([a-z0-9]+)(.*?)(/?)$",
REGEX_FLAGS_SI);
    private static final Pattern P_QUOTED_ATTRIBUTES = Pattern.compile("([a-z0-9]+)=([\"'])
(.*?)\\2", REGEX_FLAGS_SI);
    private static final Pattern P_UNQUOTED_ATTRIBUTES = Pattern.compile("([a-z0-9]+)(=)(
[^\"\\s']+)", REGEX_FLAGS_SI);
    private static final Pattern P_PROTOCOL = Pattern.compile("^([^:]+):", REGEX_FLAGS_SI);

    private static final Pattern P_ENTITY = Pattern.compile("&#(\\d+);?");
    private static final Pattern P_ENCODE = Pattern.compile("%([0-9a-f]{2});?");
    private static final Pattern P_VALID_ENTITIES = Pattern.compile("&([^&;]*)(?=(;|&|$))
");
    private static final Pattern P_VALID_QUOTES = Pattern.compile("(>|^)([^<]+?)(<|$)", P
attern.DOTALL);
    private static final Pattern P_END_ARROW = Pattern.compile("^>");
    private static final Pattern P_BODY_TO_END = Pattern.compile("<([^>]*?)(?=<|$)");
    private static final Pattern P_XML_CONTENT = Pattern.compile("(^|>)([^<]*?)(?=>)");
    private static final Pattern P_STRAY_LEFT_ARROW = Pattern.compile("<([^>]*?)(?=<|$)");
    private static final Pattern P_STRAY_RIGHT_ARROW = Pattern.compile("(^|>)([^<]*?)(?=>)"
);
    private static final Pattern P_AMP = Pattern.compile("&");
    private static final Pattern P_QUOTE = Pattern.compile("\"");
    private static final Pattern P_LEFT_ARROW = Pattern.compile("<");
    private static final Pattern P_RIGHT_ARROW = Pattern.compile(">");
    private static final Pattern P_BOTH_ARROWS = Pattern.compile("<>");
    private static final ConcurrentMap<String, Pattern> P_REMOVE_PAIR_BLANKS = new Concur
rentHashMap<>();
    private static final ConcurrentMap<String, Pattern> P_REMOVE_SELF_BLANKS = new Concur
rentHashMap<>();
    private final Map<String, List<String>> vAllowed;
    private final Map<String, Integer> vTagCounts = new HashMap<>();
    private final String[] vSelfClosingTags;
    private final String[] vNeedClosingTags;
    private final String[] vDisallowed;
    private final String[] vProtocolAtts;
    private final String[] vAllowedProtocols;
    private final String[] vRemoveBlanks;
    private final String[] vAllowedEntities;
    private final boolean stripComment;
    private final boolean encodeQuotes;
    private final boolean alwaysMakeTags;
```

```
    /**
     * Default constructor.
     */
    public HTMLFilter() {
        vAllowed = new HashMap<>();
        final ArrayList<String> a_atts = new ArrayList<>();
        a_atts.add("href");
        a_atts.add("target");
        vAllowed.put("a", a_atts);
        final ArrayList<String> img_atts = new ArrayList<>();
        img_atts.add("src");
        img_atts.add("width");
        img_atts.add("height");
        img_atts.add("alt");
        vAllowed.put("img", img_atts);
        final ArrayList<String> no_atts = new ArrayList<>();
        vAllowed.put("b", no_atts);
        vAllowed.put("strong", no_atts);
        vAllowed.put("i", no_atts);
        vAllowed.put("em", no_atts);
        vSelfClosingTags = new String[] { "img" };
        vNeedClosingTags = new String[] { "a", "b", "strong", "i", "em" };
        vDisallowed = new String[] {};
        vAllowedProtocols = new String[] { "http", "mailto", "https" }; // no ftp.
        vProtocolAtts = new String[] { "src", "href" };
        vRemoveBlanks = new String[] { "a", "b", "strong", "i", "em" };
        vAllowedEntities = new String[] { "amp", "gt", "lt", "quot" };
        stripComment = true;
        encodeQuotes = true;
        alwaysMakeTags = false;
    }
    @SuppressWarnings("unchecked")
    public HTMLFilter(final Map<String, Object> conf) {
        assert conf.containsKey("vAllowed") : "configuration requires vAllowed";
        assert conf.containsKey("vSelfClosingTags") : "configuration requires vSelfClosin
gTags";
        assert conf.containsKey("vNeedClosingTags") : "configuration requires vNeedClosin
gTags";
        assert conf.containsKey("vDisallowed") : "configuration requires vDisallowed";
        assert conf.containsKey("vAllowedProtocols") : "configuration requires vAllowedPr
otocols";
        assert conf.containsKey("vProtocolAtts") : "configuration requires vProtocolAtts";
        assert conf.containsKey("vRemoveBlanks") : "configuration requires vRemoveBlanks";
        assert conf.containsKey("vAllowedEntities") : "configuration requires vAllowedEnt
ities";

        vAllowed = Collections.unmodifiableMap((HashMap<String, List<String>>) conf.get("
vAllowed"));
        vSelfClosingTags = (String[]) conf.get("vSelfClosingTags");
        vNeedClosingTags = (String[]) conf.get("vNeedClosingTags");
```

```
        vDisallowed = (String[]) conf.get("vDisallowed");
        vAllowedProtocols = (String[]) conf.get("vAllowedProtocols");
        vProtocolAtts = (String[]) conf.get("vProtocolAtts");
        vRemoveBlanks = (String[]) conf.get("vRemoveBlanks");
        vAllowedEntities = (String[]) conf.get("vAllowedEntities");
        stripComment = conf.containsKey("stripComment") ? (Boolean) conf.get("stripCommen
t") : true;
        encodeQuotes = conf.containsKey("encodeQuotes") ? (Boolean) conf.get("encodeQuote
s") : true;
        alwaysMakeTags = conf.containsKey("alwaysMakeTags") ? (Boolean) conf.get("alwaysM
akeTags") : true;
    }

    private void reset() {
        vTagCounts.clear();
    }
    public static String chr(final int decimal) {
        return String.valueOf((char) decimal);
    }

    public static String htmlSpecialChars(final String s) {
        String result = s;
        result = regexReplace(P_AMP, "&amp;", result);
        result = regexReplace(P_QUOTE, "&quot;", result);
        result = regexReplace(P_LEFT_ARROW, "&lt;", result);
        result = regexReplace(P_RIGHT_ARROW, "&gt;", result);
        return result;
    }

    public String filter(final String input) {
        reset();
        String s = input;
        s = escapeComments(s);
        s = balanceHTML(s);
        s = checkTags(s);
        s = processRemoveBlanks(s);
        return s;
    }
    public boolean isAlwaysMakeTags() {
        return alwaysMakeTags;
    }

    public boolean isStripComments() {
        return stripComment;
    }
    private String escapeComments(final String s) {
        final Matcher m = P_COMMENTS.matcher(s);
        final StringBuffer buf = new StringBuffer();
        if (m.find()) {
            final String match = m.group(1); // (.*?)
```

```
            m.appendReplacement(buf, Matcher.quoteReplacement("<!--" + htmlSpecialChars(m
atch) + "-->"));
        }
        m.appendTail(buf);
        return buf.toString();
    }
    private String balanceHTML(String s) {
        if (alwaysMakeTags) {
            s = regexReplace(P_END_ARROW, "", s);
            s = regexReplace(P_BODY_TO_END, "<$1>", s);
            s = regexReplace(P_XML_CONTENT, "$1<$2", s);
        } else {
            s = regexReplace(P_STRAY_LEFT_ARROW, "&lt;$1", s);
            s = regexReplace(P_STRAY_RIGHT_ARROW, "$1$2&gt;<", s);
            s = regexReplace(P_BOTH_ARROWS, "", s);
        }
        return s;
    }
    private String checkTags(String s) {
        Matcher m = P_TAGS.matcher(s);
        final StringBuffer buf = new StringBuffer();
        while (m.find()) {
            String replaceStr = m.group(1);
            replaceStr = processTag(replaceStr);
            m.appendReplacement(buf, Matcher.quoteReplacement(replaceStr));
        }
        m.appendTail(buf);
        final StringBuilder sBuilder = new StringBuilder(buf.toString());
        for (String key : vTagCounts.keySet()) {
            for (int ii = 0; ii < vTagCounts.get(key); ii++) {
                sBuilder.append("</").append(key).append(">");
            }
        }
        s = sBuilder.toString();
        return s;
    }
    private String processRemoveBlanks(final String s) {
        String result = s;
        for (String tag : vRemoveBlanks) {
            if (!P_REMOVE_PAIR_BLANKS.containsKey(tag)) {
                P_REMOVE_PAIR_BLANKS.putIfAbsent(tag, Pattern.compile("<" + tag + "(\\s[^>]
*)?></" + tag + ">"));
            }
            result = regexReplace(P_REMOVE_PAIR_BLANKS.get(tag), "", result);
            if (!P_REMOVE_SELF_BLANKS.containsKey(tag)) {
                P_REMOVE_SELF_BLANKS.putIfAbsent(tag, Pattern.compile("<" + tag + "(\\s[^>]
*)?/>"));
            }
            result = regexReplace(P_REMOVE_SELF_BLANKS.get(tag), "", result);
        }
```

```
        return result;
    }
    private static String regexReplace(final Pattern regex_pattern, final String replacem
ent, final String s) {
        Matcher m = regex_pattern.matcher(s);
        return m.replaceAll(replacement);
    }
    private String processTag(final String s) {
        Matcher m = P_END_TAG.matcher(s);
        if (m.find()) {
            final String name = m.group(1).toLowerCase();
            if (allowed(name)) {
                if (!inArray(name, vSelfClosingTags)) {
                    if (vTagCounts.containsKey(name)) {
                        vTagCounts.put(name, vTagCounts.get(name) - 1);
                        return "</" + name + ">";
                    }
                }
            }
        }
        m = P_START_TAG.matcher(s);
        if (m.find()) {
            final String name = m.group(1).toLowerCase();
            final String body = m.group(2);
            String ending = m.group(3);
            if (allowed(name)) {
                final StringBuilder params = new StringBuilder();
                final Matcher m2 = P_QUOTED_ATTRIBUTES.matcher(body);
                final Matcher m3 = P_UNQUOTED_ATTRIBUTES.matcher(body);
                final List<String> paramNames = new ArrayList<>();
                final List<String> paramValues = new ArrayList<>();
                while (m2.find()) {
                    paramNames.add(m2.group(1)); // ([a-z0-9]+)
                    paramValues.add(m2.group(3)); // (.*?)
                }
                while (m3.find()) {
                    paramNames.add(m3.group(1)); // ([a-z0-9]+)
                    paramValues.add(m3.group(3)); // ([^\"\\s']+)
                }
                String paramName, paramValue;
                for (int ii = 0; ii < paramNames.size(); ii++) {
                    paramName = paramNames.get(ii).toLowerCase();
                    paramValue = paramValues.get(ii);
                    if (allowedAttribute(name, paramName)) {
                        if (inArray(paramName, vProtocolAtts)) {
                            paramValue = processParamProtocol(paramValue);
                        }
                        params.append(' ').append(paramName).append("=\\\"").append(param
Value).append("\\\"");
                    }
```

```
                }
                if (inArray(name, vSelfClosingTags)) {
                    ending = " /";
                }
                if (inArray(name, vNeedClosingTags)) {
                    ending = "";
                }
                if (ending == null || ending.length() < 1) {
                    if (vTagCounts.containsKey(name)) {
                        vTagCounts.put(name, vTagCounts.get(name) + 1);
                    } else {
                        vTagCounts.put(name, 1);
                    }
                } else {
                    ending = " /";
                }
                return "<" + name + params + ending + ">";
            } else {
                return "";
            }
        }
        m = P_COMMENT.matcher(s);
        if (!stripComment && m.find()) {
            return "<" + m.group() + ">";
        }
        return "";
    }
    private String processParamProtocol(String s) {
        s = decodeEntities(s);
        final Matcher m = P_PROTOCOL.matcher(s);
        if (m.find()) {
            final String protocol = m.group(1);
            if (!inArray(protocol, vAllowedProtocols)) {
                // bad protocol, turn into local anchor link instead
                s = "#" + s.substring(protocol.length() + 1);
                if (s.startsWith("#//")) {
                    s = "#" + s.substring(3);
                }
            }
        }
        return s;
    }
    private String decodeEntities(String s) {
        StringBuffer buf = new StringBuffer();
        Matcher m = P_ENTITY.matcher(s);
        while (m.find()) {
            final String match = m.group(1);
            final int decimal = Integer.decode(match).intValue();
            m.appendReplacement(buf, Matcher.quoteReplacement(chr(decimal)));
        }
```

```
        m.appendTail(buf);
        s = buf.toString();

        buf = new StringBuffer();
        m = P_ENTITY_UNICODE.matcher(s);
        while (m.find()) {
            final String match = m.group(1);
            final int decimal = Integer.valueOf(match, 16).intValue();
            m.appendReplacement(buf, Matcher.quoteReplacement(chr(decimal)));
        }
        m.appendTail(buf);
        s = buf.toString();

        buf = new StringBuffer();
        m = P_ENCODE.matcher(s);
        while (m.find()) {
            final String match = m.group(1);
            final int decimal = Integer.valueOf(match, 16).intValue();
            m.appendReplacement(buf, Matcher.quoteReplacement(chr(decimal)));
        }
        m.appendTail(buf);
        s = buf.toString();
        s = validateEntities(s);
        return s;
    }
    private String validateEntities(final String s) {
        StringBuffer buf = new StringBuffer();
        Matcher m = P_VALID_ENTITIES.matcher(s);
        while (m.find()) {
            final String one = m.group(1); // ([^&;]*)
            final String two = m.group(2); // (?=(;|&|$))
            m.appendReplacement(buf, Matcher.quoteReplacement(checkEntity(one, two)));
        }
        m.appendTail(buf);
        return encodeQuotes(buf.toString());
    }
    private String encodeQuotes(final String s) {
        if (encodeQuotes) {
            StringBuffer buf = new StringBuffer();
            Matcher m = P_VALID_QUOTES.matcher(s);
            while (m.find()) {
                final String one = m.group(1); // (>|^)
                final String two = m.group(2); // ([^<]+?)
                final String three = m.group(3); // (<|$)
                m.appendReplacement(buf, Matcher.quoteReplacement(one + two + three));
            }
            m.appendTail(buf);
            return buf.toString();
        } else {
            return s;
```

```
        }
    }
    private String checkEntity(final String preamble, final String term) {
        return ";".equals(term) && isValidEntity(preamble) ? '&' + preamble : "&amp;" + p
reamble;
    }
    private boolean isValidEntity(final String entity) {
        return inArray(entity, vAllowedEntities);
    }
    private static boolean inArray(final String s, final String[] array) {
        for (String item : array) {
            if (item != null && item.equals(s)) {
                return true;
            }
        }
        return false;
    }
    private boolean allowed(final String name) {
        return (vAllowed.isEmpty() || vAllowed.containsKey(name)) && !inArray(name, vDisa
llowed);
    }
    private boolean allowedAttribute(final String name, final String paramName) {
        return allowed(name) && (vAllowed.isEmpty() || vAllowed.get(name).contains(paramN
ame));
    }
}
package com.chengyu.am1ewsbd1.api.common.config.shiro;
import com.alibaba.fastjson.JSON;
import com.chengyu.am1ewsbd1.api.common.ErrorCodeEnum;
import com.chengyu.am1ewsbd1.api.common.utils.result.R;
import lombok.extern.slf4j.Slf4j;
import org.apache.commons.lang3.StringUtils;
import org.apache.shiro.authc.AuthenticationException;
import org.apache.shiro.authc.AuthenticationToken;
import org.apache.shiro.web.filter.authc.AuthenticatingFilter;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.RequestMethod;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@Slf4j
public class OAuth2Filter extends AuthenticatingFilter {

    @Override
    protected boolean isAccessAllowed(ServletRequest request, ServletResponse response, O
bject mappedValue) {
        if (((HttpServletRequest) request).getMethod().equals(RequestMethod.OPTIONS.name())
) {
            return true;
```

```
        }
        return false;
    }

    @Override
    protected boolean onAccessDenied(ServletRequest request, ServletResponse response) th
rows Exception {
        // 获取请求 token，如果 token 不存在，直接返回 401
        String token = getRequestToken((HttpServletRequest) request);
        if (StringUtils.isBlank(token)) {
            HttpServletResponse httpResponse = (HttpServletResponse) response;
            httpResponse.setHeader("Access-Control-Allow-Credentials", "true");
            httpResponse.setContentType("application/json;charset=utf-8");
            String origin = ((HttpServletRequest) request).getHeader("Origin");
            httpResponse.setHeader("Access-Control-Allow-Origin", origin);
            String json = JSON.toJSONString(R.error(ErrorCodeEnum.ADMIN_USER_INVALID_TOKE
N));
            httpResponse.getWriter().print(json);
            return false;
        }
        return executeLogin(request, response);
    }

    @Override
    protected AuthenticationToken createToken(ServletRequest request, ServletResponse res
ponse) {
        String token = getRequestToken((HttpServletRequest) request);
        if (StringUtils.isBlank(token)) {
            return null;
        }
        return new OAuth2Token(token);
    }

    @Override
    protected boolean onLoginFailure(AuthenticationToken token, AuthenticationException e,
ServletRequest request, ServletResponse response) {
        HttpServletResponse httpResponse = (HttpServletResponse) response;
        httpResponse.setContentType("application/json;charset=utf-8");
        httpResponse.setHeader("Access-Control-Allow-Credentials", "true");
        String origin = ((HttpServletRequest) request).getHeader("Origin");
        httpResponse.setHeader("Access-Control-Allow-Origin", origin);
        try {
            //处理登录失败的异常
            Throwable throwable = e.getCause() == null ? e : e.getCause();
            log.error(throwable.getMessage());
            httpResponse.setStatus(HttpStatus.UNAUTHORIZED.value());
            httpResponse.getWriter().print(JSON.toJSONString(R.error(HttpStatus.UNAUTHORI
ZED, ErrorCodeEnum.ADMIN_USER_INVALID_TOKEN).getBody()));
        } catch (Exception ex) {
            log.error(e.getMessage(), ex);
```

```
        }
        return false;
    }

    private String getRequestToken(HttpServletRequest httpRequest) {
        //从 header 中获取 token
        String token = httpRequest.getHeader("token");
        if (StringUtils.isBlank(token)) {
            token = httpRequest.getParameter("token");
        }
        return token;
    }
}
package com.chengyu.amlewsbdl.api.common.config.shiro;
import com.chengyu.amlewsbdl.api.common.constant.SystemConstant;
import com.chengyu.amlewsbdl.api.modules.admin.entity.SysUserEntity;
import com.chengyu.amlewsbdl.api.modules.admin.service.SysUserServiceImpl;
import lombok.extern.slf4j.Slf4j;
import org.apache.commons.lang3.StringUtils;
import org.apache.shiro.authc.AuthenticationException;
import org.apache.shiro.authc.AuthenticationInfo;
import org.apache.shiro.authc.AuthenticationToken;
import org.apache.shiro.authc.IncorrectCredentialsException;
import org.apache.shiro.authc.SimpleAuthenticationInfo;
import org.apache.shiro.authz.AuthorizationInfo;
import org.apache.shiro.authz.SimpleAuthorizationInfo;
import org.apache.shiro.realm.AuthorizingRealm;
import org.apache.shiro.subject.PrincipalCollection;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.redis.core.StringRedisTemplate;
import org.springframework.stereotype.Component;
import java.util.HashSet;
import java.util.Set;
/**
 * OAuth2Realm 认证实体类
 */
@Slf4j
@Component
public class OAuth2Realm extends AuthorizingRealm {

    @Autowired
    private StringRedisTemplate redisTemplate;
    @Autowired
    private SysUserServiceImpl sysUserService;

    @Override
    public boolean supports(AuthenticationToken token) {
        return token instanceof OAuth2Token;
    }
```

```
    /**
     * 授权（验证权限时调用）
     */
    @Override
    protected AuthorizationInfo doGetAuthorizationInfo(PrincipalCollection principals) {
        Set<String> permsSet = new HashSet<>();
        SimpleAuthorizationInfo info = new SimpleAuthorizationInfo();
        info.setStringPermissions(permsSet);
        return info;
    }

    /**
     * 认证
     */
    @Override
    protected AuthenticationInfo doGetAuthenticationInfo(AuthenticationToken token) throw
s AuthenticationException {
        String accessToken = (String) token.getPrincipal();
        if (StringUtils.isBlank(accessToken)) {
            throw new IncorrectCredentialsException("token 为空");
        }
        String tokenKey = SystemConstant.USER_TOKEN_REDIS_KEY + accessToken;
        String tokenCache = redisTemplate.opsForValue().get(tokenKey);
        if (StringUtils.isBlank(tokenCache)) {
            throw new IncorrectCredentialsException("token 失效，请重新登录");
        }
        String[] cacheValue = tokenCache.split("#");
        SysUserEntity user = sysUserService.getUserById(Long.parseLong(cacheValue[0]));
        if (user != null) {
            user.setAccessToken(accessToken);
            user.setSystemCode(cacheValue[1]);
            user.setRoleCode(cacheValue[2]);
            return new SimpleAuthenticationInfo(user, accessToken, getName());
        } else {
            throw new IncorrectCredentialsException("token: " + accessToken + "  未找到对
应用户");
        }
    }
}
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/myba
tis-3-mapper.dtd">
<mapper namespace="com.chengyu.amlewsbdl.mapper.TransactionFeatureMapper">
    <resultMap type="TransactionFeature" id="TransactionFeatureResult">
        <result property="featureId" column="feature_id"    />
        <result property="value" column="value"    />
        <result property="importance" column="importance"    />
        <result property="timeWindow" column="time_window"    />
        <result property="frequency" column="frequency"    />
        <result property="correlation" column="correlation"    />
```

```
        <result property="featureType" column="feature_type"    />
        <result property="anomalyScore" column="anomaly_score"    />
        <result property="transactionId" column="transaction_id"    />
        <result property="modelVersion" column="model_version"    />
    </resultMap>
    <sql id="selectTransactionFeatureVo">
        select feature_id, value, importance, time_window, frequency, correlation, featur
e_type, anomaly_score, transaction_id, model_version
        from am1ewsbd1_transactionfeature
    </sql>
    <select id="selectTransactionFeatureList" parameterType="TransactionFeature" resultMa
p="TransactionFeatureResult">
        <include refid="selectTransactionFeatureVo"/>
        <where>
            <if test="featureId != null ">
            and feature_id = #{featureId}
            </if>
            <if test="value != null and value != ''">
            and value = #{value}
            </if>
            <if test="importance != null and importance != ''">
            and importance = #{importance}
            </if>
            <if test="timeWindow != null and timeWindow != ''">
            and time_window = #{timeWindow}
            </if>
            <if test="frequency != null and frequency != ''">
            and frequency = #{frequency}
            </if>
            <if test="correlation != null and correlation != ''">
            and correlation = #{correlation}
            </if>
            <if test="featureType != null and featureType != ''">
            and feature_type = #{featureType}
            </if>
            <if test="anomalyScore != null and anomalyScore != ''">
            and anomaly_score = #{anomalyScore}
            </if>
            <if test="transactionId != null ">
            and transaction_id = #{transactionId}
            </if>
            <if test="modelVersion != null and modelVersion != ''">
            and model_version = #{modelVersion}
            </if>
        </where>
    </select>
    <delete id="deleteTransactionFeatureByFeatureId" parameterType="Integer">
        delete from am1ewsbd1_transactionfeature where feature_id = #{featureId}
    </delete>
```

```
    <insert id="insertTransactionFeature" parameterType="TransactionFeature" useGenerated
Keys="true" keyProperty="featureId">
        insert into am1ewsbd1_transactionfeature
        <trim prefix="(" suffix=")" suffixOverrides=",">
            <if test="value != null and value != ''">value,</if>
            <if test="importance != null and importance != ''">importance,</if>
            <if test="timeWindow != null and timeWindow != ''">time_window,</if>
            <if test="frequency != null and frequency != ''">frequency,</if>
            <if test="correlation != null and correlation != ''">correlation,</if>
            <if test="featureType != null and featureType != ''">feature_type,</if>
            <if test="anomalyScore != null and anomalyScore != ''">anomaly_score,</if>
            <if test="transactionId != null">transaction_id,</if>
            <if test="modelVersion != null and modelVersion != ''">model_version,</if>
         </trim>
        <trim prefix="values (" suffix=")" suffixOverrides=",">
            <if test="value != null and value != ''">#{value},</if>
            <if test="importance != null and importance != ''">#{importance},</if>
            <if test="timeWindow != null and timeWindow != ''">#{timeWindow},</if>
            <if test="frequency != null and frequency != ''">#{frequency},</if>
            <if test="correlation != null and correlation != ''">#{correlation},</if>
            <if test="featureType != null and featureType != ''">#{featureType},</if>
            <if test="anomalyScore != null and anomalyScore != ''">#{anomalyScore},</if>
            <if test="transactionId != null">#{transactionId},</if>
            <if test="modelVersion != null and modelVersion != ''">#{modelVersion},</if>
         </trim>
    </insert>
    <delete id="deleteTransactionFeatureByFeatureIds" parameterType="String">
        delete from am1ewsbd1_transactionfeature where feature_id in
        <foreach item="featureId" collection="array" open="(" separator="," close=")">
            #{featureId}
        </foreach>
    </delete>
</mapper>
package com.chengyu.am1ewsbd1.entity;
import com.baomidou.mybatisplus.annotation.IdType;
import com.baomidou.mybatisplus.annotation.TableFie1d;
import com.baomidou.mybatisplus.annotation.Table1d;
import com.baomidou.mybatisplus.annotation.TableName;
import lombok.Data;
import java.io.Serializable;
@Data
@TableName("b_error_log")
public class ErrorLog implements Serializable {
    @Table1d(value = "id",type = IdType.AUTO)
    public Long id;
    @TableFie1d
    public String ip;
    @TableFie1d
    public String url;
    @TableFie1d
```

```
    public String method;
    @TableField
    public String content;
    @TableField
    public String logTime;
}
package com.chengyu.amlewsbdl.controller;
import com.chengyu.amlewsbdl.common.APIResponse;
import com.chengyu.amlewsbdl.common.ResponeCode;
import com.chengyu.amlewsbdl.entity.ErrorLog;
import com.chengyu.amlewsbdl.service.ErrorLogService;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.transaction.annotation.Transactional;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;
import java.io.IOException;
import java.util.List;
@RestController
@RequestMapping("/errorLog")
public class ErrorLogController {

    private final static Logger logger = LoggerFactory.getLogger(ErrorLogController.class);


    @Autowired
    ErrorLogService service;

    @RequestMapping(value = "/list", method = RequestMethod.GET)
    public APIResponse list(){
        List<ErrorLog> list =  service.getErrorLogList();
        return new APIResponse(ResponeCode.SUCCESS, "查询成功", list);
    }

    @RequestMapping(value = "/create", method = RequestMethod.POST)
    @Transactional
    public APIResponse create(ErrorLog errorLog) throws IOException {
        service.createErrorLog(errorLog);
        return new APIResponse(ResponeCode.SUCCESS, "创建成功");
    }

    public Mono<Void> unsubscribe(Session session, Collection<String> gids) {
        if (gids == null || gids.isEmpty()) {
            return Mono.empty();
        }
        return Flux.fromIterable(gids)
                .flatMapSequential(gid -> {
                    var f = groupSessions.getIfPresent(gid);
```

```
                    if (f == null) {
                        return Mono.empty();
                    }
                    return Mono.fromFuture(f).doOnNext(set -> {
                        session.subGroupIds().remove(gid);
                        set.remove(session);
                        if (set.isEmpty()) {
                            groupSessions.synchronous().invalidate(gid);
                        }
                    });
                })
                .subscribeOn(Schedulers.single())
                .publishOn(Schedulers.parallel())
                .then();
    }
    @RequestMapping(value = "/delete", method = RequestMethod.POST)
    public APIResponse delete(String ids){
        System.out.println("ids===" + ids);
        // 批量删除
        String[] arr = ids.split(",");
        for (String id : arr) {
            service.deleteErrorLog(id);
        }
        return new APIResponse(ResponeCode.SUCCESS, "删除成功");
    }

    @RequestMapping(value = "/update", method = RequestMethod.POST)
    @Transactional
    public APIResponse update(ErrorLog errorLog) throws IOException {
        service.updateErrorLog(errorLog);
        return new APIResponse(ResponeCode.SUCCESS, "更新成功");
    }
}
package com.chengyu.amlewsbdl.mapper;
import java.util.List;
import com.chengyu.amlewsbdl.domain.RiskEvent;
import com.baomidou.mybatisplus.core.mapper.BaseMapper;
/**
 * 风险事件 Dao 接口
 */
public interface RiskEventMapper extends BaseMapper<RiskEvent>{
    /**
     * 新增风险事件
     *
     * @param riskEvent 风险事件
     * @return
     */
    int insertRiskEvent(RiskEvent riskEvent);

    /**
```

```
     * 修改风险事件
     *
     * @param riskEvent 风险事件
     * @return
     */
    int updateRiskEvent(RiskEvent riskEvent);

    /**
     * 查询风险事件列表
     *
     * @param eventId 风险事件主键
     * @return
     */
    RiskEvent selectRiskEventByEventId(Integer eventId);

    /**
     * 根据条件查询风险事件
     */
    List<RiskEventVo> selectByCondition(RiskEventQueryDto queryRiskEventDto);

    /**
     * 删除风险事件
     * @param eventId 风险事件主键
     * @return
     */
    int deleteRiskEventByEventId(Integer eventId);

    /**
     * 批量删除风险事件
     * @param eventIds  ID集合
     * @return
     */
    int deleteRiskEventByEventIds(Integer[] eventIds);
}

package com.chengyu.am1ewsbd1.service.imp1;
import com.baomidou.mybatisplus.extension.service.imp1.ServiceImp1;
import com.chengyu.am1ewsbd1.entity.Comment;
import com.chengyu.am1ewsbd1.mapper.CommentMapper;
import com.chengyu.am1ewsbd1.service.CommentService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;
@Service
public class CommentServiceImp1 extends ServiceImp1<CommentMapper, Comment> implements Co
mmentService {
    @Autowired
    CommentMapper mapper;

    @Override
```

```
    public List<Comment> getCommentList() {
        return mapper.getList();
    }

    @Override
    public void createComment(Comment comment) {
        System.out.println(comment);
        comment.setCommentTime(String.valueOf(System.currentTimeMillis()));
        mapper.insert(comment);
    }

    @Override
    public void deleteComment(String id) {
        mapper.deleteById(id);
    }

    @Override
    public void updateComment(Comment comment) {
        mapper.updateById(comment);
    }

    @Override
    public Comment getCommentDetail(String id) {
        return mapper.selectById(id);
    }

    @Override
    public List<Comment> getThingCommentList(String thingId, String order) {
        return mapper.selectThingCommentList(thingId, order);
    }

    @Override
    public List<Comment> getUserCommentList(String userId) {
        return mapper.selectUserCommentList(userId);
    }
}
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/myba
tis-3-mapper.dtd">
<mapper namespace="com.gk.study.mapper.CommentMapper">
    <select id="getList" resultType="com.gk.study.entity.Comment">
        select A.*, B.username, C.title
        from b_comment as A
        join b_user as B on (A.user_id=B.id)
        join b_thing as C on (A.thing_id=C.id)
        order by A.comment_time desc;
    </select>
    <select id="selectThingCommentList" parameterType="map" resultType="com.gk.study.enti
ty.Comment">
        select A.*, B.username, C.title
```

```
        from b_comment as A
                join b_user as B on (A.user_id=B.id)
                join b_thing as C on (A.thing_id=C.id)
        <if test="thingId != null">
            where A.thing_id=#{thingId}
        </if>
        <if test="order == 'recent'">
            order by A.comment_time desc;
        </if>
        <if test="order == 'hot'">
            order by A.like_count desc;
        </if>
    </select>
    <select id="selectUserCommentList" parameterType="map" resultType="com.gk.study.entit
y.Comment">
        select A.*, B.username, C.title, C.cover
        from b_comment as A
        join b_user as B on (A.user_id=B.id)
        join b_thing as C on (A.thing_id=C.id)
        <if test="userId != null">
            where A.user_id=#{userId}
        </if>
            order by A.comment_time desc;
    </select>
</mapper>
package com.chengyu.am1ewsbd1.service;
import com.chengyu.am1ewsbd1.entity.Comment;
import java.util.List;
public interface CommentService {
    List<Comment> getCommentList();
    void createComment(Comment comment);
    void deleteComment(String id);
    void updateComment(Comment comment);
    Comment getCommentDetail(String id);
    List<Comment> getThingCommentList(String thingId, String order);
    List<Comment> getUserCommentList(String userId);
}
package com.chengyu.am1ewsbd1.service.impl;
import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;
import com.chengyu.am1ewsbd1.mapper.AdMapper;
import com.chengyu.am1ewsbd1.service.AdService;
import com.chengyu.am1ewsbd1.entity.Ad;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;
@Service
public class AdServiceImpl extends ServiceImpl<AdMapper, Ad> implements AdService {
    @Autowired
    AdMapper mapper;
```

```
    @Override
    public List<Ad> getAdList() {
        return mapper.selectList(new QueryWrapper<>());
    }

    @Override
    public void createAd(Ad ad) {
        System.out.println(ad);
        ad.setCreateTime(String.valueOf(System.currentTimeMillis()));
        mapper.insert(ad);
    }

    @Override
    public void deleteAd(String id) {
        mapper.deleteById(id);
    }

    @Override
    public void updateAd(Ad ad) {
        mapper.updateById(ad);
    }
}
package com.chengyu.am1ewsbd1.service;
import com.chengyu.am1ewsbd1.entity.Address;
import java.util.List;
public interface AddressService {
    List<Address> getAddressList(String userId);
    void createAddress(Address address);
    void deleteAddress(String id);
    void updateAddress(Address address);
}
```