```
package com.chengyu.amlias.common;
import java.util.regex.Pattern;
public class RegexUtil {
    public static final Pattern TAG_PATTERN = Pattern.compile("#([^\\#|.]+)#");
    public static final String CHINESE_NAME = "^[\\u4e00-\\u9fa5]{2,32}$";
    public static final String BIRTHDAY = "([0-9]{3}[1-9]|[0-9]{2}[1-9][0-9]{1}|[0-9]{1}[
1-9][0-9]{2}|[1-9][0-9]{3})-(((0[13578]|1[02])-(0[1-9]|[12][0-9]|3[01]))|((0[469]|11)-(0[
1-9]|[12][0-9]|30))|(02-(0[1-9]|[1][0-9]|2[0-8])))";
    public static final String WXCHAT_AND_MOBILE = "^[-_a-zA-Z0-9]{5,20}$";
    public static final String PASSWORD = "^[a-zA-Z0-9]{8,15}$";
    public static final String SERIAL_NUMBER = "^[0-9].*";
    public static boolean isNumberDotStart(String input) {
        return input.matches(SERIAL_NUMBER);
    }
}

package com.chengyu.amlias.kafka_storm2;
import lombok.SneakyThrows;
import org.apache.storm.shade.org.apache.commons.collections.MapUtils;
import org.apache.storm.task.OutputCollector;
import org.apache.storm.task.TopologyContext;
import org.apache.storm.topology.OutputFieldsDeclarer;
import org.apache.storm.topology.base.BaseRichBolt;
import org.apache.storm.tuple.Tuple;
import org.example.kafka_storm2.Dao.StockListDao;
import org.example.kafka_storm2.entity.*;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.*;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
public class StatAndStoreBolt extends BaseRichBolt {
    Map<String, String[]> counters;
    Map<String, String[]> todayCounters;
    Map<String, String[]> countersStock;
    Map<String, String[]> todayCountersStock;
    Map<String, String[]> countersPlatform;
    Map<String, String[]> todayCountersPlatform;
    Map<String, String[]> countersPlace;
    Map<String, String[]> todayCountersType;
    private OutputCollector collector;
    Integer type_count = 1;
    private int processedRecords = 0;
    private long begin = System.currentTimeMillis();

    @SneakyThrows
    public void prepare(Map stormConf, TopologyContext context, OutputCollector collector)
{
        this.counters = new HashMap<String, String[]>();
```

```
        this.todayCounters = new HashMap<String, String[]>();
        this.countersStock = new HashMap<String, String[]>();
        this.todayCountersStock = new HashMap<String, String[]>();
        this.countersPlatform = new HashMap<String, String[]>();
        this.todayCountersPlatform = new HashMap<String, String[]>();
        this.countersPlace = new HashMap<String, String[]>();
        this.todayCountersType = new HashMap<String, String[]>();
        this.collector = collector;
    }
    @SneakyThrows
    public void execute(Tuple tuple) {
        Integer count = tuple.getIntegerByField("volume");
        Double amount = tuple.getDoubleByField("amount");
        String time = tuple.getStringByField("time");
        String stockCode = tuple.getStringByField("stockCode");
        String stockType = tuple.getStringByField("tradeType");
        String stockName = tuple.getStringByField("stockName");
        String tradePlace = tuple.getStringByField("tradePlace");
        String tradePlatform = tuple.getStringByField("tradePlatform");
        String industryType = tuple.getStringByField("industryType");
        processedRecords++;
        long current = System.currentTimeMillis();
        if (current - begin >= 1000) {
            System.out.println("Records processed per second: " + processedRecords);
            StockListDao.UpdateCount(processedRecords);
            processedRecords = 0;
            begin = current;
        }
        try {
            String[] s = new String[2];
            String t = time.split(":")[0]+":"+time.split(":")[1] + "_" + stockType;
            if(!counters.containsKey(t)){
                Minite minite = new Minite();
                minite.setMiniteName(t);
                minite.setMiniteTime(time);
                minite.setTotalCount(count);
                minite.setTotalAmount(amount);
                minite.setStockName(stockName);
                minite.setStockType(stockType);
                minite.setStockCode(stockCode);
                minite.setPrice(amount);
                minite.setPlace(tradePlace);
                minite.setPlatform(tradePlatform);
                minite.setIndustryType(industryType);
                StockListDao.InsertMinite(minite);
                s[0] = String.valueOf(count);
                s[1] = String.valueOf(amount);
                counters.put(t, s);
            }else{
                String[] strings = counters.get(t);
```

```
                long count2 = Integer.parseInt(strings[0]) + count;
                double amount2 = Double.parseDouble(strings[1]) + amount;
                s[0] = String.valueOf(count2);
                s[1] = String.valueOf(amount2);
                counters.put(t, s);
                StockListDao.UpdateMinite(t, count2, amount2);
            }
            collector.ack(tuple);
        } catch (Exception e) {
            e.printStackTrace();
            collector.fail(tuple);
        }
        try {
            String[] s = new String[2];
            //2023-11-14 10:46:45
            String dayTime = time.split(" ")[0] + stockType;
            if(!todayCounters.containsKey(dayTime)){
                Day day = new Day();
                day.setDayName(dayTime);
                day.setDayTime(time);
                day.setTotalCount(count);
                day.setTotalAmount(amount);
                day.setStockName(stockName);
                day.setStockType(stockType);
                day.setStockCode(stockCode);
                day.setPrice(amount);
                day.setPlace(tradePlace);
                day.setPlatform(tradePlatform);
                day.setIndustryType(industryType);

                StockListDao.InsertDay(day);

                s[0] = String.valueOf(count);
                s[1] = String.valueOf(amount);
                todayCounters.put(dayTime, s);
            }else{
                String[] strings = todayCounters.get(dayTime);
                long count2 = Integer.parseInt(strings[0]) + count;
                double amount2 = Double.parseDouble(strings[1]) + amount;
                s[0] = String.valueOf(count2);
                s[1] = String.valueOf(amount2);
                todayCounters.put(dayTime, s);
                StockListDao.UpdateDay(dayTime, count2, amount2);
            }
            collector.ack(tuple);
        } catch (Exception e) {
            e.printStackTrace();
        }
        try {
            String[] s = new String[2];
```

```
            //2023-11-14 10:46:45
            String dayTime = time.split(" ")[0] + stockName;
            if(!todayCountersStock.containsKey(dayTime)){
                Daystock daystock = new Daystock();
                daystock.setDayName(dayTime);
                daystock.setDayTime(time);
                daystock.setTotalCount(count);
                daystock.setTotalAmount(amount);
                daystock.setStockName(stockName);
                daystock.setStockCode(stockCode);
                daystock.setPrice(amount);
                daystock.setIndustryType(industryType);
                StockListDao.InsertDayStock(daystock);
                s[0] = String.valueOf(count);
                s[1] = String.valueOf(amount);
                todayCountersStock.put(dayTime, s);
            }else{
                String[] strings = todayCountersStock.get(dayTime);
                long count2 = Integer.parseInt(strings[0]) + count;
                double amount2 = Double.parseDouble(strings[1]) + amount;
                s[0] = String.valueOf(count2);
                s[1] = String.valueOf(amount2);
                todayCountersStock.put(dayTime, s);
                StockListDao.UpdateDayStock(dayTime, count2, amount2);
            }
            collector.ack(tuple);
        } catch (Exception e) {
            e.printStackTrace();
        }
        try {
            String[] s = new String[2];
            //2023-11-14 10:46:45
            String t = time.split(":")[0]+":"+time.split(":")[1] + "_" + stockName;

            if(!countersStock.containsKey(t)){
                Minitestock minitestock = new Minitestock();
                minitestock.setMiniteName(t);
                minitestock.setMiniteTime(time);
                minitestock.setTotalCount(count);
                minitestock.setTotalAmount(amount);
                minitestock.setStockName(stockName);
                minitestock.setStockCode(stockCode);
                minitestock.setPrice(amount);
                minitestock.setIndustryType(industryType);

                StockListDao.InsertMiniteStock(minitestock);

                s[0] = String.valueOf(count);
                s[1] = String.valueOf(amount);
                countersStock.put(t, s);
```

```
            }else{
                String[] strings = countersStock.get(t);
                long count2 = Integer.parseInt(strings[0]) + count;
                double amount2 = Double.parseDouble(strings[1]) + amount;
                s[0] = String.valueOf(count2);
                s[1] = String.valueOf(amount2);
                countersStock.put(t, s);
                StockListDao.UpdateMiniteStock(t, count2, amount2);
            }
            collector.ack(tuple);
        } catch (Exception e) {
            e.printStackTrace();
        }
        try {
            String[] s = new String[2];
            //2023-11-14 10:46:45
            String t = time.split(":")[0]+":"+time.split(":")[1] + "_" + tradePlatform;
            if(!countersPlatform.containsKey(t)){
                Miniteplatform miniteplatform = new Miniteplatform();
                miniteplatform.setMiniteName(t);
                miniteplatform.setMiniteTime(time);
                miniteplatform.setTotalCount(count);
                miniteplatform.setTotalAmount(amount);
                miniteplatform.setPlatform(tradePlatform);
                StockListDao.InsertMiniteplatform(miniteplatform);
                s[0] = String.valueOf(count);
                s[1] = String.valueOf(amount);
                countersPlatform.put(t, s);
            }else{
                String[] strings = countersPlatform.get(t);
                long count2 = Integer.parseInt(strings[0]) + count;
                double amount2 = Double.parseDouble(strings[1]) + amount;
                s[0] = String.valueOf(count2);
                s[1] = String.valueOf(amount2);
                countersPlatform.put(t, s);
                StockListDao.UpdateMiniteplatform(t, count2, amount2);
            }
            collector.ack(tuple);
        } catch (Exception e) {
            e.printStackTrace();
        }
        try {
            String[] s = new String[2];
            //2023-11-14 10:46:45
            String t = time.split(" ")[0]+ tradePlatform;

            if(!todayCountersPlatform.containsKey(t)){
                Dayplatform dayplatform = new Dayplatform();
                dayplatform.setDayName(t);
                dayplatform.setDayTime(time);
```

```
            dayplatform.setTotalCount(count);
            dayplatform.setTotalAmount(amount);
            dayplatform.setPlatform(tradePlatform);
            StockListDao.InsertDayplatform(dayplatform);
            s[0] = String.valueOf(count);
            s[1] = String.valueOf(amount);
            todayCountersPlatform.put(t, s);
        }else{
            String[] strings = todayCountersPlatform.get(t);
            long count2 = Integer.parseInt(strings[0]) + count;
            double amount2 = Double.parseDouble(strings[1]) + amount;
            s[0] = String.valueOf(count2);
            s[1] = String.valueOf(amount2);
            todayCountersPlatform.put(t, s);
            StockListDao.UpdateDayplatform(t, count2, amount2);
        }
        collector.ack(tuple);
    } catch (Exception e) {
        e.printStackTrace();
    }
    try {
        String[] s = new String[2];
        Pattern pattern = Pattern.compile("(.*?)(省|市)");
        Matcher matcher = pattern.matcher(tradePlace);
        if (matcher.find()) {
            tradePlace = matcher.group(1);
        }
        if(!countersPlace.containsKey(tradePlace)){
            Place place = new Place();
            place.setPlaceName(tradePlace);
            place.setTotalCount(count);
            place.setTotalAmount(count*amount);
            StockListDao.InsertPlace(place);
            s[0] = String.valueOf(count);
            s[1] = String.valueOf(count*amount);
            countersPlace.put(tradePlace, s);
        }else{
            String[] strings = countersPlace.get(tradePlace);
            long count2 = Integer.parseInt(strings[0]) + count;
            double amount2 = Double.parseDouble(strings[1]) + count*amount;
            s[0] = String.valueOf(count2);
            s[1] = String.valueOf(amount2);
            countersPlace.put(tradePlace, s);
            StockListDao.UpdatePlace(tradePlace, count2, amount2);
        }
        collector.ack(tuple);
    } catch (Exception e) {
        e.printStackTrace();
    }
    try {
```

```
            String[] s = new String[2];
            String name = industryType;
            if(!todayCountersType.containsKey(name)){
                Type type = new Type();
                type.setTypeName(name);
                type.setTotalCount(count);
                type.setTotalAmount(amount);
                type.setPlatform(tradePlatform);
                type.setPlace(tradePlace);
                StockListDao.InsertType(type);
                s[0] = String.valueOf(count);
                s[1] = String.valueOf(amount);

                todayCountersType.put(name, s);
            }else{
                String[] strings = todayCountersType.get(name);
                long count2 = Integer.parseInt(strings[0]) + count;
                double amount2 = Double.parseDouble(strings[1]) + amount;
                s[0] = String.valueOf(count2);
                s[1] = String.valueOf(amount2);
                todayCountersType.put(name, s);
                StockListDao.UpdateType(name, count2, amount2);
            }
            collector.ack(tuple);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public void declareOutputFields(OutputFieldsDeclarer declarer) {
    }
}
<?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.amlias.mapper.TransactionMapper">
    <resultMap type="Transaction" id="TransactionResult">
        <result property="transactionId" column="transaction_ID"    />
        <result property="updatedAt" column="updated_at"    />
        <result property="transactionDate" column="transaction_date"    />
        <result property="amount" column="amount"    />
        <result property="currency" column="currency"    />
        <result property="status" column="status"    />
        <result property="suspicionLevel" column="suspicion_level"    />
        <result property="source" column="source"    />
        <result property="reported" column="reported"    />
        <result property="reportDate" column="report_date"    />
        <result property="description" column="description"    />
        <result property="destination" column="destination"    />
        <result property="createdAt" column="created_at"    />
        <result property="accountId" column="account_ID"    />
```

```
        <result property="machineLearningScore" column="machine_learning_score"    />
        <result property="transactionType" column="transaction_type"    />
    </resultMap>
    <sql id="selectTransactionVo">
        select transaction_ID, updated_at, transaction_date, amount, currency, status, su
spicion_level, source, reported, report_date, description, destination, created_at, accou
nt_ID, machine_learning_score, transaction_type
        from aml_transaction
    </sql>
    <select id="selectTransactionList" parameterType="Transaction" resultMap="Transaction
Result">
        <include refid="selectTransactionVo"/>
        <where>
            <if test="transactionId != null ">
            and transaction_ID = #{transactionId}
            </if>
            <if test="updatedAt != null ">
            and updated_at = #{updatedAt}
            </if>
            <if test="transactionDate != null and transactionDate != ''">
            and transaction_date = #{transactionDate}
            </if>
            <if test="amount != null ">
            and amount = #{amount}
            </if>
            <if test="currency != null and currency != ''">
            and currency = #{currency}
            </if>
            <if test="status != null and status != ''">
            and status = #{status}
            </if>
            <if test="suspicionLevel != null ">
            and suspicion_level = #{suspicionLevel}
            </if>
            <if test="source != null and source != ''">
            and source = #{source}
            </if>
            <if test="reported != null ">
            and reported = #{reported}
            </if>
            <if test="reportDate != null and reportDate != ''">
            and report_date = #{reportDate}
            </if>
            <if test="description != null and description != ''">
            and description = #{description}
            </if>
            <if test="destination != null and destination != ''">
            and destination = #{destination}
            </if>
            <if test="createdAt != null ">
```

```
            and created_at = #{createdAt}
            </if>
            <if test="accountId != null ">
            and account_ID = #{accountId}
            </if>
            <if test="machineLearningScore != null ">
            and machine_learning_score = #{machineLearningScore}
            </if>
            <if test="transactionType != null and transactionType != ''">
            and transaction_type = #{transactionType}
            </if>
        </where>
    </select>
    <delete id="deleteTransactionByTransactionId" parameterType="Integer">
        delete from amlias_transaction where transaction_ID = #{transactionId}
    </delete>

    <insert id="insertTransaction" parameterType="Transaction" useGeneratedKeys="true" ke
yProperty="transactionId">
        insert into amlias_transaction
        <trim prefix="(" suffix=")" suffixOverrides=",">
            <if test="updatedAt != null">updated_at,</if>
            <if test="transactionDate != null and transactionDate != ''">transaction_date,<
/if>
            <if test="amount != null">amount,</if>
            <if test="currency != null and currency != ''">currency,</if>
            <if test="status != null and status != ''">status,</if>
            <if test="suspicionLevel != null">suspicion_level,</if>
            <if test="source != null and source != ''">source,</if>
            <if test="reported != null">reported,</if>
            <if test="reportDate != null and reportDate != ''">report_date,</if>
            <if test="description != null and description != ''">description,</if>
            <if test="destination != null and destination != ''">destination,</if>
            <if test="createdAt != null">created_at,</if>
            <if test="accountId != null">account_ID,</if>
            <if test="machineLearningScore != null">machine_learning_score,</if>
            <if test="transactionType != null and transactionType != ''">transaction_type,<
/if>
         </trim>
        <trim prefix="values (" suffix=")" suffixOverrides=",">
            <if test="updatedAt != null">#{updatedAt},</if>
            <if test="transactionDate != null and transactionDate != ''">#{transactionDat
e},</if>
            <if test="amount != null">#{amount},</if>
            <if test="currency != null and currency != ''">#{currency},</if>
            <if test="status != null and status != ''">#{status},</if>
            <if test="suspicionLevel != null">#{suspicionLevel},</if>
            <if test="source != null and source != ''">#{source},</if>
            <if test="reported != null">#{reported},</if>
            <if test="reportDate != null and reportDate != ''">#{reportDate},</if>
```

```
            <if test="description != null and description != ''">#{description},</if>
            <if test="destination != null and destination != ''">#{destination},</if>
            <if test="createdAt != null">#{createdAt},</if>
            <if test="accountId != null">#{accountId},</if>
            <if test="machineLearningScore != null">#{machineLearningScore},</if>
            <if test="transactionType != null and transactionType != ''">#{transactionTyp
e},</if>
         </trim>
    </insert>
    <delete id="deleteTransactionByTransactionIds" parameterType="String">
        delete from amlias_transaction where transaction_ID in
        <foreach item="transactionId" collection="array" open="(" separator="," close=")">
            #{transactionId}
        </foreach>
    </delete>
</mapper>
package com.chengyu.amlias.mapper;
import java.util.List;
import com.chengyu.amlias.domain.InvestigationRecord;
import com.baomidou.mybatisplus.core.mapper.BaseMapper;
/**
 * 调查记录 Dao 接口
 */
public interface InvestigationRecordMapper extends BaseMapper<InvestigationRecord>{
    /**
     * 新增调查记录
     *
     * @param investigationRecord 调查记录
     * @return
     */
    int insertInvestigationRecord(InvestigationRecord investigationRecord);

    /**
     * 修改调查记录
     *
     * @param investigationRecord 调查记录
     * @return
     */
    int updateInvestigationRecord(InvestigationRecord investigationRecord);

    /**
     * 查询调查记录列表
     *
     * @param recordId 调查记录主键
     * @return
     */
    InvestigationRecord selectInvestigationRecordByRecordId(Integer recordId);

    /**
     * 根据条件查询调查记录
```

```
     */
    List<InvestigationRecordVo> selectByCondition(InvestigationRecordQueryDto queryInvest
igationRecordDto);

    /**
     * 删除调查记录
     * @param recordId 调查记录主键
     * @return
     */
    int deleteInvestigationRecordByRecordId(Integer recordId);

    /**
     * 批量删除调查记录
     * @param recordIds  ID集合
     * @return
     */
    int deleteInvestigationRecordByRecordIds(Integer[] recordIds);
}

package com.chengyu.amlias.modules.admin.service;
import com.alibaba.fastjson.JSONArray;
import com.alibaba.fastjson.JSONObject;
import com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;
import com.baomidou.mybatisplus.core.conditions.update.LambdaUpdateWrapper;
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;
import com.chengyu.amlias.common.constant.SystemConstant;
import com.chengyu.amlias.common.exception.GlobalException;
import com.chengyu.amlias.modules.admin.dao.SysMenuDao;
import com.chengyu.amlias.modules.admin.entity.SysMenuEntity;
import com.chengyu.amlias.modules.admin.entity.SysRoleMenuEntity;
import com.chengyu.amlias.modules.admin.model.SysMenuTree;
import com.chengyu.amlias.modules.admin.model.TreeModel;
import org.apache.commons.lang3.StringUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;
@Service
public class SysMenuServiceImpl extends ServiceImpl<SysMenuDao, SysMenuEntity> {

    @Autowired
    private SysRoleMenuServiceImpl sysRoleMenuService;

    public void addPermission(SysMenuEntity sysPermission) {
        if (SystemConstant.MenuTypeEnum.MENU_TYPE_0.getValue() == sysPermission.getMenuTy
```

```
pe()) {
            sysPermission.setParentId(0L);
        } else {
            if (sysPermission.getParentId() == null) {
                throw new GlobalException("父级菜单 ID 参数不存在");
            } else {
                SysMenuEntity menu = this.getById(sysPermission.getParentId());
                if (menu == null) {
                    throw new GlobalException("所选父级数据不存在");
                }
            }
        }
        Long parentId = sysPermission.getParentId();
        if (parentId != null && parentId != 0) {
            this.update(new LambdaUpdateWrapper<SysMenuEntity>().eq(SysMenuEntity::getId,
parentId).set(SysMenuEntity::getIsLeaf, 0));
        }
        sysPermission.setIsLeaf(1);
        this.save(sysPermission);
    }

    public void editPermission(SysMenuEntity sysPermission) {
        SysMenuEntity oldP = this.getById(sysPermission.getId());
        if (oldP == null) {
            throw new GlobalException("未找到菜单信息");
        } else {
            if (SystemConstant.MenuTypeEnum.MENU_TYPE_0.getValue() == sysPermission.getMe
nuType()) {
                sysPermission.setParentId(0L);
            }
            int count = this.count(new LambdaQueryWrapper<SysMenuEntity>().eq(SysMenuEnti
ty::getParentId, sysPermission.getId()));
            if (count == 0) {
                sysPermission.setIsLeaf(1);
            }
            this.updateById(sysPermission);
            Long pid = sysPermission.getParentId();
            if (!pid.equals(oldP.getParentId())) {
                this.update(new LambdaUpdateWrapper<SysMenuEntity>().eq(SysMenuEntity::ge
tId, pid).set(SysMenuEntity::getIsLeaf, 0));
                int cc = this.count(new LambdaQueryWrapper<SysMenuEntity>().eq(SysMenuEnt
ity::getParentId, oldP.getParentId()));
                if (cc == 0) {
                    this.update(new LambdaUpdateWrapper<SysMenuEntity>().eq(SysMenuEntity::
getId, oldP.getParentId()).set(SysMenuEntity::getIsLeaf, 1));
                }
            }
        }
    }
```

```
    /**
     * 删除信息
     *
     * @param id 记录 ID
     */
    @Transactional
    public void deletePermission(Long id) {
        SysMenuEntity sysPermission = this.getById(id);
        if (sysPermission == null) {
            throw new GlobalException("未找到信息");
        }
        Long pid = sysPermission.getParentId();
        int count = this.count(new LambdaQueryWrapper<SysMenuEntity>().eq(SysMenuEntity::
getParentId, pid));
        if (count == 1) {
            this.update(new LambdaUpdateWrapper<SysMenuEntity>().eq(SysMenuEntity::getId,
pid).set(SysMenuEntity::getIsLeaf, 1));
        }
        this.removeById(id);
        this.removeChildrenBy(sysPermission.getId());
        sysRoleMenuService.remove(new LambdaQueryWrapper<SysRoleMenuEntity>().eq(SysRoleM
enuEntity::getMenuId, id));
    }

    public JSONObject getUserPermission(String username, String systemCode) {
        List<SysMenuEntity> metaList = this.baseMapper.getUserPermission(username, system
Code);
        if (metaList.size() == 0) {
            throw new GlobalException("用户对应角色未分配权限");
        }
        JSONObject json = new JSONObject();
        JSONArray menuDataArray = new JSONArray();
        this.getPermissionJsonArray(menuDataArray, metaList, null);
        json.put("menu", menuDataArray);
        List<String> auth =  metaList.stream().filter(f -> f.getMenuType().equals(SystemC
onstant.MenuTypeEnum.MENU_TYPE_2.getValue()))
                .map(SysMenuEntity::getPermsCode).collect(Collectors.toList());
        json.put("button", auth);
        return json;
    }

    public Map<String, Object> queryTreeList(String systemCode) {
        LambdaQueryWrapper<SysMenuEntity> query = new LambdaQueryWrapper<>();
        query.eq(SysMenuEntity::getSystemCode, systemCode).orderByAsc(SysMenuEntity::getS
ortNo);
        List<SysMenuEntity> menuList = this.list(query);
        List<Long> ids = new ArrayList<>();
        Map<Long, List<TreeModel>> treeList = menuList.stream().peek(p -> ids.add(p.getId
())).map(TreeModel::new).collect(Collectors.groupingBy(TreeModel::getParentId));
        List<TreeModel> root = treeList.get(0L);
```

```
        circulateMenuParse(root, treeList);
        Map<String, Object> resMap = new HashMap<>();
        resMap.put("treeList", root);
        resMap.put("ids", ids);
        return resMap;
    }

    public void circulateMenuParse(List<TreeModel> root, Map<Long, List<TreeModel>> group
ByParent) {
        for (TreeModel temp : root) {
            List<TreeModel> child = groupByParent.get(temp.getKey());
            if (child == null) {
                continue;
            }
            temp.setChildren(child);
            circulateMenuParse(child, groupByParent);
        }
    }

    public Map<Long, List<SysMenuTree>> getSystemSubmenuBatch(List<Long> parentIdList) {
        List<SysMenuEntity> list = this.list(new LambdaQueryWrapper<SysMenuEntity>().in(S
ysMenuEntity::getParentId, parentIdList).orderByAsc(SysMenuEntity::getSortNo));
        Map<Long, List<SysMenuTree>> listMap = new HashMap<>();
        for (SysMenuEntity item : list) {
            List<SysMenuTree> mapList = listMap.getOrDefault(item.getParentId(), new Arra
yList<>());
            mapList.add(new SysMenuTree(item));
            listMap.put(item.getParentId(), mapList);
        }
        return listMap;
    }

    public Boolean checkRolePermission(String permissionIds) {
        List<Long> newPermissionIds = Arrays.stream(permissionIds.split(",")).map(Long::p
arseLong).collect(Collectors.toList());
        for (Long permissionId : newPermissionIds) {
            SysMenuEntity permission = this.getById(permissionId);
            if (permission.getParentId() != 0) {
                if (!newPermissionIds.contains(permission.getParentId())) {
                    return false;
                }
            }
        }
        return true;
    }

    private void getPermissionJsonArray(JSONArray jsonArray, List<SysMenuEntity> metaList,
JSONObject parentJson) {
        for (SysMenuEntity permission : metaList) {
            if (permission.getMenuType() == null || permission.getMenuType().equals(Syste
```

```
mConstant.MenuTypeEnum.MENU_TYPE_2.getValue())) {
                continue;
            }
            JSONObject json = getPermissionJsonObject(permission);
            Long tempPid = permission.getParentId();
            if (parentJson == null && tempPid == 0) {
                jsonArray.add(json);
                if (permission.getIsLeaf().equals(0)) {
                    getPermissionJsonArray(jsonArray, metaList, json);
                }
            } else if (parentJson != null && tempPid != 0 && tempPid.equals(parentJson.ge
tLong("id"))) {
                if (parentJson.containsKey("children")) {
                    parentJson.getJSONArray("children").add(json);
                } else {
                    JSONArray children = new JSONArray();
                    children.add(json);
                    parentJson.put("children", children);
                }
                if (permission.getIsLeaf().equals(0)) {
                    getPermissionJsonArray(jsonArray, metaList, json);
                }
            }
        }
    }

    private JSONObject getPermissionJsonObject(SysMenuEntity permission) {
        JSONObject json = new JSONObject();
        json.put("id", permission.getId());
        json.put("path", permission.getUrl());
        json.put("name", urlToRouteName(permission.getUrl()));
        json.put("component", permission.getComponent());
        JSONObject meta = new JSONObject();
        meta.put("title", permission.getName());
        meta.put("icon", permission.getIcon());
        meta.put("parentPath", permission.getParentUrl());
        meta.put("isHidden", permission.getIsHidden().equals(1));
        meta.put("showAlways", permission.getShowAlways().equals(1));
        String component = permission.getComponent();
        if (StringUtils.isNotBlank(component)) {
            meta.put("componentName", component.substring(component.lastIndexOf("/") + 1));

        }
        if (permission.getParentId() == 0) {
            json.put("redirect", permission.getRedirect());
        }
        json.put("meta", meta);
        return json;
    }
```

```
    private String urlToRouteName(String url) {
        if (StringUtils.isNotEmpty(url)) {
            if (url.startsWith("/")) {
                url = url.substring(1);
            }
            url = url.replace("/", "-");
            url = url.replace(":", "@");
            return url;
        } else {
            return null;
        }
    }

    /**
     * 根据 id 删除其关联的子数据
     */
    public void removeChildrenBy(Long parentId) {
        LambdaQueryWrapper<SysMenuEntity> query = new LambdaQueryWrapper<>();
        query.eq(SysMenuEntity::getParentId, parentId);
        List<SysMenuEntity> permissionList = this.list(query);
        if (permissionList != null && permissionList.size() > 0) {
            Long id; // id
            int num;
            this.remove(query);
            for (SysMenuEntity sysMenuEntity : permissionList) {
                id = sysMenuEntity.getId();
                sysRoleMenuService.remove(new LambdaQueryWrapper<SysRoleMenuEntity>().eq(
SysRoleMenuEntity::getMenuId, id));
                num = this.count(new LambdaQueryWrapper<SysMenuEntity>().eq(SysMenuEntity::
getParentId, id));
                if (num > 0) {
                    this.removeChildrenBy(id);
                }
            }
        }
    }
}
package com.chengyu.amlias.service.impl;
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;
import com.chengyu.amlias.mapper.AccountMapper;
import com.chengyu.amlias.domain.Account;
import com.chengyu.amlias.service.IAccountService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;
@Service
public class AccountServiceImpl extends ServiceImpl<AccountMapper, Account> implements IA
ccountService{
    @Autowired
    private AccountMapper accountMapper;
```

```
@Override
public Account getAccountByAccountId(Integer accountId){
    return accountMapper.selectAccountByAccountId(accountId);
}

@Override
public List<Account> listAccount(Account account){
    return accountMapper.selectAccountList(account);
}

@Transactional
@Override
public int saveAccount(Account account){
    return accountMapper.insertAccount(account);
}

private void checkExisted(AccountEntity entity) {
    AccountEntity existEntity = getOne(
            Wrappers.lambdaQuery(AccountEntity.class)
            .eq(AccountEntity::getAccountId, entity.getAccountId)
            false);
    if (existEntity != null && !existEntity.getId().equals(entity.getId())) {
        throw new BizException("账户信息记录已存在！");
    }
}

@Override
@Transactional
public int delAccountByAccountIds(Integer[] accountIds){
    return accountMapper.deleteAccountByAccountIds(accountIds);
}

private void accountData(List<AccountDto> list) {
    List<String> list = list.stream().map(AccountDto::getAccountId)
            .collect(Collectors.toList());
    list.stream().forEach(account -> {
        Account accountEntry = entryAccountMap.get(account.getAccountId);
        if (accountEntry != null) {
            account.setCustomerId(AccountTrans(accountEntry.getCustomerId));
            account.setMonitorReason(AccountTrans(accountEntry.getMonitorReason));
            account.setCloseDate(AccountTrans(accountEntry.getCloseDate));
            account.setAccountId(AccountTrans(accountEntry.getAccountId));
            account.setAccountNumber(AccountTrans(accountEntry.getAccountNumber));
        }
    });
}

@Override
@Transactional
```

```
    public int updateAccount(Account account){
        return accountMapper.updateAccount(account);
    }

    @Override
    @Transactional
    public int delAccountByAccountId(Integer accountId){
        return accountMapper.deleteAccountByAccountId(accountId);
    }
}
package com.chengyu.amlias.domain;
import java.math.BigDecimal;
import java.util.Date;
import com.fasterxml.jackson.annotation.JsonFormat;
import lombok.Data;
import com.chengyu.amlias.common.annotation.Excel;
import com.baomidou.mybatisplus.annotation.TableName;
import com.chengyu.amlias.common.core.domain.BaseEntity;
/**
 * 交易记录实体类
 */
@TableName("amlias_transaction")
@Data
public class Transaction extends BaseEntity {
    private static final long serialVersionUID = -1652537082245114157LL;
    // 交易 ID
    private Integer transactionId;
    // 更新时间
    private Date updatedAt;
    // 交易日期
    private String transactionDate;
    // 交易金额
    private BigDecimal amount;
    // 货币类型
    private String currency;
    // 交易状态
    private String status;
    // 可疑程度等级
    private Integer suspicionLevel;
    // 交易来源
    private String source;
    // 是否已报告
    private Integer reported;
    // 报告日期
    private String reportDate;
    // 交易描述
    private String description;
    // 交易目的地
    private String destination;
    // 创建时间
```

```
    private Date createdAt;
    // 账户 ID
    private Integer accountId;
    // 机器学习评分
    private BigDecimal machineLearningScore;
    // 交易类型
    private String transactionType;
}
package com.chengyu.aml.modules.admin.controller;
import com.chengyu.aml.common.ErrorCodeEnum;
import com.chengyu.aml.common.annotation.SysLog;
import com.chengyu.aml.common.exception.GlobalException;
import com.chengyu.aml.common.utils.HardwareUtils;
import com.chengyu.aml.common.utils.result.R;
import com.chengyu.aml.modules.admin.service.SysLicenseServiceImpl;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiImplicitParam;
import io.swagger.annotations.ApiImplicitParams;
import io.swagger.annotations.ApiOperation;
import io.swagger.annotations.ApiParam;
import lombok.extern.slf4j.Slf4j;
import org.apache.commons.io.IOUtils;
import org.apache.commons.lang3.StringUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpHeaders;
import org.springframework.http.MediaType;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.multipart.MultipartFile;
import javax.servlet.http.HttpServletResponse;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.nio.charset.StandardCharsets;
@Slf4j
@RestController
@RequestMapping("/sys/license")
@Api(tags = "系统授权")
public class SysLicenseSettingController {

    @Autowired
    private SysLicenseServiceImpl sysLicenseService;

    @SuppressWarnings("ResultOfMethodCallIgnored")
    @GetMapping("/download")
    @ApiOperation("下载服务器硬件信息")
    @SysLog(value = "下载服务器硬件信息", isSaveDatabase = true)
    public void downLoadLicense(HttpServletResponse response) {
```

```
        String filePath = HardwareUtils.initLicense();
        File file = new File(filePath);
        if (!file.exists()) {
            throw new GlobalException("下载服务器硬件信息异常");
        }
        response.setContentType(MediaType.APPLICATION_OCTET_STREAM_VALUE);
        response.setCharacterEncoding(StandardCharsets.UTF_8.name());
        response.setContentLength((int) file.length());
        response.setHeader(HttpHeaders.CONTENT_DISPOSITION, "attachment;filename=license.
info");
        try {
            IOUtils.copy(new FileInputStream(file), response.getOutputStream());
            response.flushBuffer();
        } catch (Exception e) {
            throw new GlobalException("下载服务器硬件信息异常");
        } finally {
            file.delete();
        }
    }

    @PostMapping("/upload")
    @ApiOperation("上传服务器程序 License 信息")
    @SysLog(value = "上传服务器程序 License 信息", isSaveDatabase = true)
    public R uploadLicense(@ApiParam("License 文件") MultipartFile file, String licenseCon
tent) throws IOException {
        if (StringUtils.isBlank(licenseContent)) {
            if (file == null) {
                throw new GlobalException("未发现上传的 License 信息");
            }
            licenseContent = IOUtils.toString(file.getInputStream(), StandardCharsets.UTF
_8.name());
        }
        if (HardwareUtils.hardwareNotMatch(licenseContent)) { // 硬件信息匹配判断
            return R.successCode(ErrorCodeEnum.LICENSE_VALID_EXCEPTION);
        }
        // 保存信息到数据库
        return sysLicenseService.updateLicenseInfo(licenseContent);
    }

    @GetMapping("/info")
    @ApiOperation("获取 License 信息")
    public R searchLicenseInfo() {
        return R.success(sysLicenseService.searchLicenseInfo());
    }

    @GetMapping("/check_license")
    @ApiOperation("验证 License 合法性")
    public R CheckLicenseInfo() {
        return sysLicenseService.checkLicenseStatus();
    }
```

```
}
package com.chengyu.aml1as.modules.admin.controller;
import com.baomidou.mybatisplus.core.metadata.IPage;
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;
import com.chengyu.aml1as.common.annotation.SysLog;
import com.chengyu.aml1as.common.utils.result.R;
import com.chengyu.aml1as.modules.admin.entity.SysLogEntity;
import com.chengyu.aml1as.modules.admin.service.SysLogServiceImpl;
import com.chengyu.aml1as.modules.admin.vo.ReqSysLogVo;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiImplicitParam;
import io.swagger.annotations.ApiImplicitParams;
import io.swagger.annotations.ApiOperation;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;
@Slf4j
@RestController
@RequestMapping("/sys/log")
@Api(tags = {"系统日志"})
public class SysLogController {

    @Autowired
    private SysLogServiceImpl sysLogService;

    @ApiOperation("系统日志-分页")
    @SysLog(value = "系统日志分页列表查询")
    @GetMapping
    @ApiImplicitParams({
            @ApiImplicitParam(name = "pageNo", value = "当前页", example = "1"),
            @ApiImplicitParam(name = "pageSize", value = "每页大小", example = "10")
    })
    public R getSysLogPageList(
            ReqSysLogVo reqSysLogVo,
            @RequestParam(name = "pageNo", defaultValue = "1") Integer pageNo,
            @RequestParam(name = "pageSize", defaultValue = "10") Integer pageSize) {
        IPage<SysLogEntity> page = new Page<>(pageNo, pageSize);
        return R.success(sysLogService.getSysLogPageList(page, reqSysLogVo));
    }

    @ApiOperation("系统日志-删除")
    @SysLog(value = "删除系统操作日志", isSaveDatabase = true)
    @DeleteMapping
```

```
    public R deleteSysLog(String ids) {
        List<Long> idList = Arrays.stream(ids.split(",")).map(Long::parseLong).collect(Co
llectors.toList());
        sysLogService.removeByIds(idList);
        return R.success();
    }
}
package com.chengyu.am1ias.common.vo;
import lombok.Data;
import lombok.experimental.Accessors;
import java.util.List;
import java.util.Map;
@Accessors(chain = true)
@Data
public class SearchResultVo<T> {

    // 返回的记录集合
    private List<T> records;

    // 返回的拓展集合
    private Map<?, ?> maps;

    // 当前页
    private Integer current;

    // 总条数
    private Integer total;

    // 每页大小
    private Integer size;

    /**
     * 总页数
     */
    private Integer pages;

    public SearchResultVo(Integer current, Integer size, Integer total) {
        if (current != null && size != null) {
            this.current = current;
            this.total = total;
            this.size = size;
            this.setPages(total % size == 0 ? total / size : total / size + 1);
        }
    }

    public SearchResultVo(List<T> records, Integer current, Integer total, Integer size)
{
        this.records = records;
        this.current = current;
        this.total = total;
```

```
        this.size = size;
        if (current != null && size != null) {
            this.setPages(total % size == 0 ? total / size : total / size + 1);
        }
    }

    public SearchResultVo(List<T> records, Map<?, ?> maps, Integer current, Integer total,
Integer size) {
        this.records = records;
        this.maps = maps;
        this.current = current;
        this.total = total;
        this.size = size;
        if (current != null && size != null) {
            this.setPages(total % size == 0 ? total / size : total / size + 1);
        }
    }

    public SearchResultVo() {
    }
}
package com.chengyu.amlias.common.utils.result;
import com.alibaba.fastjson.annotation.JSONField;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Data;
@Data
@ApiModel(value = "状态")
public class ResultStatus {

    @ApiModelProperty("状态码")
    @JSONField(ordinal = 1)
    private int code;

    @ApiModelProperty("信息描述")
    @JSONField(ordinal = 2)
    private String message;
}
package com.chengyu.amlias.modules.admin.controller;
import com.chengyu.amlias.common.ErrorCodeEnum;
import com.chengyu.amlias.common.utils.result.R;
import com.chengyu.amlias.modules.admin.dao.DuplicateCheckDao;
import com.chengyu.amlias.modules.admin.vo.system.DuplicateCheckVo;
import lombok.extern.slf4j.Slf4j;
import org.apache.commons.lang3.StringUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;
```

```
@Slf4j
@RestController
@RequestMapping("/sys/duplicate")
public class DuplicateCheckController {
    @Autowired
    private DuplicateCheckDao duplicateCheckDao;
    @RequestMapping(value = "/check", method = RequestMethod.GET)
    public R doDuplicateCheck(DuplicateCheckVo duplicateCheckVo) {
        Long num;
        // 关联表字典（举例：system_user,realname,id）
        Long delFlagFieldCount = duplicateCheckDao.delFlagFieldCheck(duplicateCheckVo);
        // 判断表中是否存在逻辑删除字段，如果存在那么需要对逻辑删除字段进行判断，逻辑删除字段为“is_delete”(写死)
        if (delFlagFieldCount == 0) {
            if (StringUtils.isNotBlank(duplicateCheckVo.getDataId())) {
                // [2].编辑页面校验
                num = duplicateCheckDao.duplicateCheckCountSql(duplicateCheckVo);
            } else {
                // [1].添加页面校验
                num = duplicateCheckDao.duplicateCheckCountSqlNoDataId(duplicateCheckVo);
            }
        } else {
            if (StringUtils.isNotBlank(duplicateCheckVo.getDataId())) {
                // [2].编辑页面校验
                num = duplicateCheckDao.duplicateCheckWithDelFlag(duplicateCheckVo);
            } else {
                // [1].添加页面校验
                num = duplicateCheckDao.duplicateCheckNoDataIdWithDelFlag(duplicateCheckVo);
            }
        }
        if (num == null || num == 0) {
            return R.success("该值可用");
        } else {
            return R.error(HttpStatus.OK, ErrorCodeEnum.FAILED.getCode(), "该值已存在");
        }
    }
}
package com.chengyu.amlias.domain;
import java.util.Date;
import com.fasterxml.jackson.annotation.JsonFormat;
import lombok.Data;
import com.chengyu.amlias.common.annotation.Excel;
import com.baomidou.mybatisplus.annotation.TableName;
import com.chengyu.amlias.common.core.domain.BaseEntity;
/**
 * 预警记录实体类
 */
@TableName("amlias_alertrecord")
@Data
```

```
public class AlertRecord extends BaseEntity {
    private static final long serialVersionUID = -2720771415916660211LL;
    // 预警 ID
    private Integer alertId;
    // 预警时间
    private String alertTime;
    // 创建时间
    private Date createdAt;
    // 调查状态
    private String investigationStatus;
    // 预警信息
    private String alertMessage;
    // 更新时间
    private Date updatedAt;
    // 关联的交易 ID
    private Integer transactionId;
    // 触发的监控规则 ID
    private Integer ruleId;
    // 预警级别
    private String alertLevel;
}
<?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.amlias.mapper.MonitoringRuleMapper">
    <resultMap type="MonitoringRule" id="MonitoringRuleResult">
        <result property="ruleId" column="rule_ID"    />
        <result property="updatedAt" column="updated_at"    />
        <result property="createdAt" column="created_at"    />
        <result property="triggerCondition" column="trigger_condition"    />
        <result property="description" column="description"    />
        <result property="ruleName" column="rule_name"    />
        <result property="active" column="active"    />
    </resultMap>
    <sql id="selectMonitoringRuleVo">
        select rule_ID, updated_at, created_at, trigger_condition, description, rule_name,
active
        from amlias_monitoringrule
    </sql>
    <select id="selectMonitoringRuleList" parameterType="MonitoringRule" resultMap="Monit
oringRuleResult">
        <include refid="selectMonitoringRuleVo"/>
        <where>
            <if test="ruleId != null ">
            and rule_ID = #{ruleId}
            </if>
            <if test="updatedAt != null ">
            and updated_at = #{updatedAt}
            </if>
            <if test="createdAt != null ">
```

```
            and created_at = #{createdAt}
            </if>
            <if test="triggerCondition != null and triggerCondition != ''">
            and trigger_condition = #{triggerCondition}
            </if>
            <if test="description != null and description != ''">
            and description = #{description}
            </if>
            <if test="ruleName != null and ruleName != ''">
            and rule_name = #{ruleName}
            </if>
            <if test="active != null ">
            and active = #{active}
            </if>
        </where>
    </select>
    <delete id="deleteMonitoringRuleByRuleId" parameterType="Integer">
        delete from amlias_monitoringrule where rule_ID = #{ruleId}
    </delete>

    <insert id="insertMonitoringRule" parameterType="MonitoringRule" useGeneratedKeys="tr
ue" keyProperty="ruleId">
        insert into amlias_monitoringrule
        <trim prefix="(" suffix=")" suffixOverrides=",">
            <if test="updatedAt != null">updated_at,</if>
            <if test="createdAt != null">created_at,</if>
            <if test="triggerCondition != null and triggerCondition != ''">trigger_condit
ion,</if>
            <if test="description != null and description != ''">description,</if>
            <if test="ruleName != null and ruleName != ''">rule_name,</if>
            <if test="active != null">active,</if>
         </trim>
        <trim prefix="values (" suffix=")" suffixOverrides=",">
            <if test="updatedAt != null">#{updatedAt},</if>
            <if test="createdAt != null">#{createdAt},</if>
            <if test="triggerCondition != null and triggerCondition != ''">#{triggerCondi
tion},</if>
            <if test="description != null and description != ''">#{description},</if>
            <if test="ruleName != null and ruleName != ''">#{ruleName},</if>
            <if test="active != null">#{active},</if>
         </trim>
    </insert>
    <delete id="deleteMonitoringRuleByRuleIds" parameterType="String">
        delete from amlias_monitoringrule where rule_ID in
        <foreach item="ruleId" collection="array" open="(" separator="," close=")">
            #{ruleId}
        </foreach>
    </delete>
</mapper>
package com.chengyu.amlias.domain;
```

```
import java.math.BigDecimal;
import java.util.Date;
import com.fasterxml.jackson.annotation.JsonFormat;
import lombok.Data;
import com.chengyu.aml ias.common.annotation.Excel;
import com.baomidou.mybatisplus.annotation.TableName;
import com.chengyu.aml ias.common.core.domain.BaseEntity;
/**
 * 机器学习模型实体类
 */
@TableName("aml ias_ml model")
@Data
public class MLModel extends BaseEntity {
    private static final long serialVersionUID = -4240740043072791782LL;
    // 模型 ID
    private Integer modelId;
    // 更新时间
    private Date updatedAt;
    // 模型类型
    private String modelType;
    // 训练数据描述
    private String trainingData;
    // 模型名称
    private String modelName;
    // 创建时间
    private Date createdAt;
    // 准确率
    private BigDecimal accuracy;
}
package com.chengyu.aml ias.controller;
import java.util.List;
import javax.servlet.http.HttpServletResponse;
import com.chengyu.aml ias.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.aml ias.common.annotation.Log;
import com.chengyu.aml ias.common.core.controller.BaseController;
import com.chengyu.aml ias.common.utils.poi.ExcelUtil;
import com.chengyu.aml ias.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.aml ias.common.enums.BusinessType;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import com.chengyu.aml ias.domain.MonitoringRule;
import com.chengyu.aml ias.service.IMonitoringRuleService;
import org.springframework.web.bind.annotation.PathVariable;
```

```
@RestController
@RequestMapping("/monitoringrule")
public class MonitoringRuleController extends BaseAction {
    @Autowired
    private IMonitoringRuleService monitoringRuleService;

    /**
     * 查询监控规则列表
     */
    @GetMapping("/list")
    public TableDataInfo monitoringRuleList(MonitoringRuleQueryVo queryVo){
        startPage();
        List<MonitoringRule> monitoringRuleList = monitoringRuleService.getMonitoringRule
List(queryVo);
        list.forEach(i -> {
            i.setDescription(MonitoringRuleConvUtil(i.getDescription));
            i.setTriggerCondition(MonitoringRuleConvUtil(i.getTriggerCondition));
        });
        PageInfo<MonitoringRule> pageInfo = new PageInfo<>(list);
        return RespResult.pageResult(list, pageInfo.getTotal());
    }

    @PostMapping
    public RespResult addMonitoringRule(@Valid @RequestBody MonitoringRuleAddVo addVo){
       monitoringRuleService.saveMonitoringRule(addVo);
       return RespResult.success();
    }

    public RespResult listmonitoringRule(MonitoringRule monitoringRule){
        List<MonitoringRule> list = monitoringRuleService.getMonitoringRuleList(monitorin
gRule);
        return RespResult.success(list);
    }

    @DeleteMapping("/{ruleIds}")
    public RespResult delmonitoringRule(@PathVariable Integer[] ruleIds){
        monitoringRuleService.deleteMonitoringRuleByRuleIds(ruleIds);
        return RespResult.success();
    }

    @PutMapping
    public RespResult updatemonitoringRule(@Valid @RequestBody MonitoringRule monitoringR
ule){
        monitoringRuleService.updateMonitoringRule(monitoringRule);
        return RespResult.success();
    }

    @PostMapping("/export")
    public void exportMonitoringRule(HttpServletResponse response, MonitoringRule monitor
ingRule){
```

```
        List<MonitoringRule> list = monitoringRuleService.getMonitoringRuleList(monitorin
gRule);
        list.forEach(i -> {
            i.setDescription(MonitoringRuleExportFormat(i.getDescription));
            i.setTriggerCondition(MonitoringRuleExportFormat(i.getTriggerCondition));
        });
        ExcelUtil<MonitoringRule> excelUtil = new ExcelUtil<MonitoringRule>(MonitoringRul
e.class);
        util.exportExcel(response, list, "监控规则数据");
    }

    @GetMapping(value = "/{ruleId}")
    public RespResult getruleId(@PathVariable("ruleId") Integer ruleId){
        return RespResult.success(monitoringRuleService.getMonitoringRuleByRuleId(ruleId));

    }
}
package com.chengyu.amlias.entity;
import com.baomidou.mybatisplus.annotation.IdType;
import com.baomidou.mybatisplus.annotation.TableField;
import com.baomidou.mybatisplus.annotation.TableId;
import com.baomidou.mybatisplus.annotation.TableName;
import com.fasterxml.jackson.annotation.JsonFormat;
import com.fasterxml.jackson.annotation.JsonIgnore;
import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
import lombok.Data;
import org.springframework.web.multipart.MultipartFile;
import java.io.Serializable;
@Data
@TableName("b_user")
public class User implements Serializable {
    public static final int NormalUser = 1;
    public static final int DemoUser = 2;
    public static final int AdminUser = 3;

    @TableId(value = "id",type = IdType.AUTO)
    public String id;
    @TableField
    public String username;
    @TableField
    public String password;
    @TableField(exist = false)
    public String rePassword;
    @TableField
    public String nickname;
    @TableField
    public String mobile;
    @TableField
    public String email;
    @TableField
```

```
    public String description;
    @TableField
    public String role;
    @TableField
    public String status;
    @TableField
    public String score;
    @TableField
    public String avatar;
    @TableField(exist = false)
    public MultipartFile avatarFile;
    @TableField
    public String token;
    @TableField
    public String createTime;
    @TableField
    public String pushEmail;
    @TableField
    public String pushSwitch;
}

package com.chengyu.amlias.controller;
import com.chengyu.amlias.common.APIResponse;
import com.chengyu.amlias.common.ResponeCode;
import com.chengyu.amlias.entity.User;
import com.chengyu.amlias.permission.Access;
import com.chengyu.amlias.permission.AccessLevel;
import com.chengyu.amlias.service.UserService;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.transaction.annotation.Transactional;
import org.springframework.util.DigestUtils;
import org.springframework.util.StringUtils;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;
import java.io.File;
import java.io.IOException;
import java.util.List;
import java.util.UUID;
/**
 * 登录注册
 */
@RestController
@RequestMapping("/user")
public class UserController {
    private final static Logger logger = LoggerFactory.getLogger(UserController.class);
    String salt = "abc_dl%23#4";
```

```
    @Autowired
    UserService userService;

    @Value("${File.uploadPath}")
    private String uploadPath;

    @RequestMapping(value = "/list", method = RequestMethod.GET)
    public APIResponse list(String keyword){
        List<User> list =  userService.getUserList(keyword);
        return new APIResponse(ResponeCode.SUCCESS, "查询成功", list);
    }

    @RequestMapping(value = "/detail", method = RequestMethod.GET)
    public APIResponse detail(String userId){
        User user =  userService.getUserDetail(userId);
        return new APIResponse(ResponeCode.SUCCESS, "查询成功", user);
    }

    // 后台用户登录
    @RequestMapping(value = "/login", method = RequestMethod.POST)
    public APIResponse login(User user){
        user.setPassword(DigestUtils.md5DigestAsHex((user.getPassword() + salt).getBytes())
);
        User responseUser =  userService.getAdminUser(user);
        if(responseUser != null) {
            return new APIResponse(ResponeCode.SUCCESS, "查询成功", responseUser);
        }else {
            return new APIResponse(ResponeCode.FAIL, "用户名或密码错误");
        }
    }

    // 普通用户登录
    @RequestMapping(value = "/userLogin", method = RequestMethod.POST)
    public APIResponse userLogin(User user){
        user.setPassword(DigestUtils.md5DigestAsHex((user.getPassword() + salt).getBytes())
);
        User responseUser =  userService.getNormalUser(user);
        if(responseUser != null) {
            return new APIResponse(ResponeCode.SUCCESS, "查询成功", responseUser);
        }else {
            return new APIResponse(ResponeCode.FAIL, "用户名或密码错误");
        }
    }

    // 普通用户注册
    @RequestMapping(value = "/userRegister", method = RequestMethod.POST)
    @Transactional
    public APIResponse userRegister(User user) throws IOException {
        if (!StringUtils.isEmpty(user.getUsername())
                && !StringUtils.isEmpty(user.getPassword())
```

```
                && !StringUtils.isEmpty(user.getRePassword())) {
            // 查重
            if(userService.getUserByUserName(user.getUsername()) != null) {
                return new APIResponse(ResponeCode.FAIL, "用户名重复");
            }
            // 验证密码
            if(!user.getPassword().equals(user.getRePassword())) {
                return new APIResponse(ResponeCode.FAIL, "密码不一致");
            }
            String md5Str = DigestUtils.md5DigestAsHex((user.getPassword() + salt).getByt
es());
            // 设置密码
            user.setPassword(md5Str);
            md5Str = DigestUtils.md5DigestAsHex((user.getUsername() + salt).getBytes());
            // 设置 token
            user.setToken(md5Str);

            String avatar = saveAvatar(user);
            if(!StringUtils.isEmpty(avatar)) {
                user.avatar = avatar;
            }
            // 设置角色
            user.setRole(String.valueOf(User.NormalUser));
            // 设置状态
            user.setStatus("0");
            user.setCreateTime(String.valueOf(System.currentTimeMillis()));
            userService.createUser(user);
            return new APIResponse(ResponeCode.SUCCESS, "创建成功");
        }
        return new APIResponse(ResponeCode.FAIL, "创建失败");
    }

    @Access(level = AccessLevel.ADMIN)
    @RequestMapping(value = "/create", method = RequestMethod.POST)
    @Transactional
    public APIResponse create(User user) throws IOException {
        if (!StringUtils.isEmpty(user.getUsername()) || !StringUtils.isEmpty(user.getPass
word())) {
            // 查重
            if(userService.getUserByUserName(user.getUsername()) != null) {
                return new APIResponse(ResponeCode.FAIL, "用户名重复");
            }
            String md5Str = DigestUtils.md5DigestAsHex((user.getPassword() + salt).getByt
es());
            // 设置密码
            user.setPassword(md5Str);
            md5Str = DigestUtils.md5DigestAsHex((user.getUsername() + salt).getBytes());
            // 设置 token
            user.setToken(md5Str);
            user.setCreateTime(String.valueOf(System.currentTimeMillis()));
```

```
            String avatar = saveAvatar(user);
            if(!StringUtils.isEmpty(avatar)) {
                user.avatar = avatar;
            }
            userService.createUser(user);
            return new APIResponse(ResponeCode.SUCCESS, "创建成功");
        }
        return new APIResponse(ResponeCode.FAIL, "创建失败");
    }

    @Access(level = AccessLevel.ADMIN)
    @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) {
            userService.deleteUser(id);
        }
        return new APIResponse(ResponeCode.SUCCESS, "删除成功");
    }

    @Access(level = AccessLevel.ADMIN)
    @RequestMapping(value = "/update", method = RequestMethod.POST)
    @Transactional
    public APIResponse update(User user) throws IOException {
        // update 不能修改密码，故置空
        user.setPassword(null);
        String avatar = saveAvatar(user);
        if(!StringUtils.isEmpty(avatar)) {
            user.avatar = avatar;
        }
        userService.updateUser(user);
        System.out.println(user);
        return new APIResponse(ResponeCode.SUCCESS, "更新成功");
    }

    @Access(level = AccessLevel.LOGIN)
    @RequestMapping(value = "/updateUserInfo", method = RequestMethod.POST)
    @Transactional
    public APIResponse updateUserInfo(User user) throws IOException {
        User tmpUser =  userService.getUserDetail(user.getId());
        if(tmpUser.getRole().equals(String.valueOf(User.NormalUser))){
            // username 和 password 不能改，故置空
            user.setUsername(null);
            user.setPassword(null);
            user.setRole(String.valueOf(User.NormalUser));
            String avatar = saveAvatar(user);
            if(!StringUtils.isEmpty(avatar)) {
```

```
                user.avatar = avatar;
            }
            userService.updateUser(user);
            return new APIResponse(ResponeCode.SUCCESS, "更新成功");
        }else {
            return new APIResponse(ResponeCode.FAIL, "非法操作");
        }
    }

    @Access(level = AccessLevel.LOGIN)
    @RequestMapping(value = "/updatePwd", method = RequestMethod.POST)
    @Transactional
    public APIResponse updatePwd(String userId, String password, String newPassword) thro
ws IOException {
        User user =  userService.getUserDetail(userId);
        if(user.getRole().equals(String.valueOf(User.NormalUser))) {
            String md5Pwd = DigestUtils.md5DigestAsHex((password + salt).getBytes());
            if(user.getPassword().equals(md5Pwd)){
                user.setPassword(DigestUtils.md5DigestAsHex((newPassword + salt).getBytes
()));
                userService.updateUser(user);
            }else {
                return new APIResponse(ResponeCode.FAIL, "原密码错误");
            }
            return new APIResponse(ResponeCode.SUCCESS, "更新成功");
        }else {
            return new APIResponse(ResponeCode.FAIL, "非法操作");
        }

    }

    public String saveAvatar(User user) throws IOException {
        MultipartFile file = user.getAvatarFile();
        String newFileName = null;
        if(file !=null && !file.isEmpty()) {
            // 存文件
            String oldFileName = file.getOriginalFilename();
            String randomStr = UUID.randomUUID().toString();
            newFileName = randomStr + oldFileName.substring(oldFileName.lastIndexOf("."));
            String filePath = uploadPath + File.separator + "avatar" + File.separator + n
ewFileName;
            File destFile = new File(filePath);
            if(!destFile.getParentFile().exists()){
                destFile.getParentFile().mkdirs();
            }
            file.transferTo(destFile);
        }
        if(!StringUtils.isEmpty(newFileName)) {
            user.avatar = newFileName;
        }
```

```
        return newFileName;
    }
}
package com.chengyu.amlias.modules.admin.controller;
import com.chengyu.amlias.common.annotation.SysLog;
import com.chengyu.amlias.common.utils.result.RetResponse;
import com.chengyu.amlias.modules.admin.service.SysProgramServiceImpl;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiImplicitParam;
import io.swagger.annotations.ApiImplicitParams;
import io.swagger.annotations.ApiOperation;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import javax.annotation.Resource;
@RestController
@RequestMapping("/sys/service")
@Api(tags = "系统服务")
public class SysProgramServiceController {

    @Resource
    private SysProgramServiceImpl sysProgramService;

    @SysLog(value = "基础服务状态获取", isSaveDatabase = true)
    @ApiOperation(value = "基础服务状态获取")
    @GetMapping(value = "/status")
    public RetResponse searchServiceStatus() {
        return RetResponse.success(sysProgramService.getServiceStatus());
    }

    @SysLog(value = "重启服务", isSaveDatabase = true)
    @ApiOperation(value = "重启服务")
    @ApiImplicitParams({
            @ApiImplicitParam(paramType = "query", name = "serviceType", value = "服务类
型, 如 data/mysql", dataType = "String")
    })
    @GetMapping(value = "/reboot")
    public RetResponse searchTimeSetting(@RequestParam String serviceType) {
        boolean success = sysProgramService.restartService(serviceType);
        return success ? RetResponse.success() : RetResponse.error("重启失败");
    }

    @SysLog(value = "系统资源获取", isSaveDatabase = true)
    @ApiOperation(value = "系统资源获取")
    @GetMapping(value = "/resource")
    public RetResponse searchSystemResource() {
        return RetResponse.success(sysProgramService.systemResource());
    }
}
```

```
package com.chengyu.amlias.modules.admin.controller;
import com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;
import com.chengyu.amlias.common.annotation.SysLog;
import com.chengyu.amlias.common.constant.SystemConstant;
import com.chengyu.amlias.common.utils.OAuth2UserUtils;
import com.chengyu.amlias.common.utils.result.RetResponse;
import com.chengyu.amlias.common.utils.result.RunLog;
import com.chengyu.amlias.modules.admin.entity.SysRoleEntity;
import com.chengyu.amlias.modules.admin.service.SysRoleServiceImpl;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import io.swagger.annotations.ApiParam;
import lombok.extern.slf4j.Slf4j;
import org.apache.commons.lang3.StringUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;
@Slf4j
@RestController
@RequestMapping("/sys/role")
@Api(tags = "角色管理")
public class SysRoleController {

    @Autowired
    private SysRoleServiceImpl sysRoleService;

    @SysLog(value = "角色列表", isSaveDatabase = true)
    @ApiOperation(value = "角色列表", notes = "角色列表，分页展示")
    @RequestMapping(value = "/list", method = RequestMethod.GET)
    public RetResponse queryPageList(
            SysRoleEntity role,
            @RequestParam(name = "pageNo", defaultValue = "1") Integer pageNo,
            @RequestParam(name = "pageSize", defaultValue = "10") Integer pageSize) {
        return RetResponse.success(sysRoleService.searchRoleListPage(role, pageNo, pageSi
ze));
    }

    @SysLog(value = "添加角色", isSaveDatabase = true)
    @ApiOperation(value = "添加角色")
    @RequestMapping(value = "/add", method = RequestMethod.POST)
    public RetResponse add(@RequestBody SysRoleEntity role) {
        try {
            role.setCreateBy(OAuth2UserUtils.getOauthUserName());
            sysRoleService.save(role);
```

```
            return RetResponse.success();
        } catch (Exception e) {
            log.error(e.getMessage(), e.fillInStackTrace());
            return RetResponse.error("操作失败");
        }
    }

    @SysLog(value = "编辑角色", isSaveDatabase = true)
    @ApiOperation(value = "编辑角色")
    @RequestMapping(value = "/edit", method = RequestMethod.PUT)
    public RetResponse edit(@RequestBody SysRoleEntity role) {
        SysRoleEntity sysRole = sysRoleService.getById(role.getId());
        if (sysRole == null) {
            return RetResponse.error("角色不存在");
        }
        if (sysRole.getSysDefault() == SystemConstant.SysDefaultEnum.DEFAULT_0.getCode())
{
            return RetResponse.error("系统默认角色，不能修改");
        }
        boolean ok = sysRoleService.updateById(role);
        if (ok) {
            return RetResponse.success();
        }
        return RetResponse.error();
    }

    @SysLog(value = "删除角色", isSaveDatabase = true)
    @ApiOperation(value = "删除角色", notes = "通过 id 删除角色")
    @RequestMapping(value = "/delete", method = RequestMethod.DELETE)
    public RetResponse delete(@ApiParam(value = "角色 id") @RequestParam(name = "id") Long
id) {
        try {
            sysRoleService.deleteRole(id);
            return RetResponse.success();
        } catch (Exception e) {
            log.error(e.getMessage(), e.fillInStackTrace());
            return RetResponse.error(e.getMessage());
        }
    }

    @SysLog(value = "批量删除", isSaveDatabase = true, readRunLog = true)
    @ApiOperation(value = "批量删除", notes = "批量删除角色")
    @RequestMapping(value = "/deleteBatch", method = RequestMethod.DELETE)
    public RetResponse deleteBatch(@ApiParam(value = "角色 ids,多个 id 之间用逗号分割") @Re
questParam(name = "ids") String ids) {
        if (StringUtils.isEmpty(ids)) {
            return RetResponse.error("未选中角色！");
        }
        List<Long> idList = Arrays.stream(ids.split(",")).map(Long::parseLong).collect(Co
llectors.toList());
```

```
        List<SysRoleEntity> list = sysRoleService.list(new LambdaQueryWrapper<SysRoleEnti
ty>()
                .select(SysRoleEntity::getId, SysRoleEntity::getRoleName).in(SysRoleEntit
y::getId, idList));
        if (list != null && list.size() > 0) {
            sysRoleService.deleteBatchRole(list.stream().map(SysRoleEntity::getId).collec
t(Collectors.toList()));
        }
        return RetResponse.successWithLog(RunLog.create(list));
    }

    @SysLog(value = "查询所有的角色", isSaveDatabase = true)
    @ApiOperation(value = "查询所有的角色")
    @RequestMapping(value = "/queryall", method = RequestMethod.GET)
    public RetResponse queryAll(@RequestParam(name = "systemCode", defaultValue = "002")
String systemCode) {
        return RetResponse.success(sysRoleService.selectRoleList(systemCode));
    }

    @SysLog(value = "查询角色所属类型列表", isSaveDatabase = true)
    @ApiOperation(value = "查询角色所属类型列表")
    @RequestMapping(value = "/search_role_classify", method = RequestMethod.GET)
    public RetResponse searchRoleClassify() {
        return RetResponse.success(sysRoleService.searchRoleClassifyList());
    }
}
package com.chengyu.amlias.common.base.result;
import io.swagger.v3.oas.annotations.media.Schema;
import lombok.Data;
import java.util.HashMap;
import java.util.Map;
@Data
public class Ret {
    @Schema(description = "是否成功")
    private Boolean success;

    @Schema(description = "返回码")
    private Integer code;

    @Schema(description = "返回消息")
    private String message;

    @Schema(description = "返回数据")
    private Map<String, Object> data = new HashMap<String, Object>();

    private Ret(){}

    public static Ret ok(){
        Ret r = new Ret();
        r.setSuccess(ResultCode.SUCCESS.getSuccess());
```

```
        r.setCode(RetesultCode.SUCCESS.getCode());
        r.setMessage(RetesultCode.SUCCESS.getMessage());
        return r;
    }

    public static Ret error(){
        Ret r = new Ret();
        r.setSuccess(false);
        r.setCode(RetesultCode.UNKNOWN_RetEASON.getCode());
        r.setMessage("失败");
        return r;
    }

    public static Ret setRetesult(RetesultCode resultCode) {
        Ret r = new Ret();
        r.setSuccess(resultCode.getSuccess());
        r.setCode(resultCode.getCode());
        r.setMessage(resultCode.getMessage());
        return r;
    }

    public Ret success(Boolean success){
        this.setSuccess(success);
        return this;
    }

    public Ret message(String message){
        this.setMessage(message);
        return this;
    }

    public Ret code(Integer code){
        this.setCode(code);
        return this;
    }

    public Ret data(String key, Object value){
        this.data.put(key, value);
        return this;
    }

    public Ret data(Map<String, Object> map){
        this.setData(map);
        return this;
    }
}
package com.chengyu.amlias.common.base.result;
import lombok.Getter;
import lombok.Setter;
@Getter
```

```
public enum RetesultCode {
    SUCCESS(true, 20000, "成功"),
    UNNOWN_RetEASON(false, 20001, "未知错误"),
    BAD_SQL_GRetAMMARet(false, 21001, "sql 语法错误"),
    JSON_PARetSE_ERetRetORet(false, 21002, "json 解析异常"),
    PARetAM_ERetRetORet(false, 21003, "参数不正确"),
    FILE_UPLOAD_ERetRetORet(false, 21004, "文件上传错误"),
    FILE_DELETE_ERetRetORet(false, 21005, "文件删除错误"),
    EXCEL_DATA_IMPORetT_ERetRetORet(false, 21006, "Excel 数据导入错误"),
    VIDEO_UPLOAD_ALIYUN_ERetRetORet(false, 22001, "文件上传阿里云失败"),
    VIDEO_UPLOAD_TOMCAT_ERetRetORet(false, 22002, "文件上传 tomcat 失败"),
    VIDEO_DELETE_ALIYUN_ERetRetORet(false, 22003, "阿里云文件视频删除失败"),
    FETCH_VIDEO_UPLOADAUTH_ERetRetORet(false, 22004, "获取上传地址和凭证失败"),
    FETCH_PLAYAUTH_ERetRetORet(false, 22005, "获取播放凭证失败"),
    URetL_ENCODE_ERetRetORet(false, 23001, "URetL 编码失败"),
    ILLEGAL_CALLBACK_RetEQUEST_ERetRetORet(false, 23002, "非法回调请求"),
    FETCH_ACCESS_TOKEN_FAILED(false, 23003, "获取 accessToken 失败"),
    FETCH_USERetINFO_ERetRetORet(false, 23004, "获取用户信息失败"),
    LOGIN_ERetRetORet(false, 23005, "登录失败"),
    COMMENT_EMPTH(false, 24006, "评论内容必须填写"),
    PAY_RetUN(false, 25000, "支付中"),
    PAY_UNIFIED_ERetRetORet(false, 25001, "统一下单错误"),
    PAY_ORetDERetQUERetY_ERetRetORet(false, 25002, "查询支付结果错误"),
    ORetDERet_EXIST_ERetRetORet(false, 25003, "课程已购买"),
    GATEWAY_ERetRetORet(false, 26000, "服务无法访问"),
    CODE_ERetRetORet(false, 28000, "验证码错误"),
    LOGIN_PHONE_ERetRetORet(false, 28001, "手机号码不正确"),
    LOGIN_ACCOUNT_ERetRetORet(false, 28002, "账号不正确"),
    LOGIN_PASSWORetD_ERetRetORet(false, 28008, "密码不正确"),
    LOGIN_DISABLED_ERetRetORet(false, 28004, "该账号已禁用"),
    RetEGISTERet_MOBLE_ERetRetORet(false, 28005, "手机号已注册"),
    RetEGISTERet_NICKNAME_ERetRetORet(false, 28006, "昵称重复"),
    LOGIN_AUTH(false, 28007, "需要登录"),
    LOGIN_ACL(false, 28008, "没有权限"),
    SMS_SEND_ERetRetORet(false, 28009, "短信发送失败"),
    SMS_SEND_ERetRetORet_BUSINESS_LIMIT_CONTRetOL(false, 28010, "短信发送频繁");
    private Boolean success;
    private Integer code;
    private String message;
    RetesultCode(Boolean success, Integer code, String message) {
        this.success = success;
        this.code = code;
        this.message = message;
    }
}
package com.chengyu.amlias.mapper;
import java.util.List;
import com.chengyu.amlias.domain.Account;
import com.baomidou.mybatisplus.core.mapper.BaseMapper;
/**
```

```
 * 账户信息 Dao 接口
 */
public interface AccountMapper extends BaseMapper<Account>{
    /**
     * 新增账户信息
     *
     * @param account 账户信息
     * @return
     */
    int insertAccount(Account account);

    /**
     * 修改账户信息
     *
     * @param account 账户信息
     * @return
     */
    int updateAccount(Account account);

    /**
     * 查询账户信息列表
     *
     * @param accountId 账户信息主键
     * @return
     */
    Account selectAccountByAccountId(Integer accountId);

    /**
     * 根据条件查询账户信息
     */
    List<AccountVo> selectByCondition(AccountQueryDto queryAccountDto);

    /**
     * 删除账户信息
     * @param accountId 账户信息主键
     * @return
     */
    int deleteAccountByAccountId(Integer accountId);

    /**
     * 批量删除账户信息
     * @param accountIds  ID 集合
     * @return
     */
    int deleteAccountByAccountIds(Integer[] accountIds);
}

package com.chengyu.amlias.common.readcsv;
import org.apache.flume.Event;
import org.apache.flume.event.EventBuilder;
```

```
import java.io.IOException;
import java.nio.file.*;
import java.io.*;
import java.util.HashMap;
import java.util.Map;
@Service
@Slf4j
public class WatchServiceResp {
    public static void watch() {
        try {
            File dir = new File("/home/tmp/wath10");
            Path dirPath = dir.toPath();
            WatchService watchService = FileSystems.getDefault().newWatchService();
            dirPath.register(watchService, StandardWatchEventKinds.ENTRY_CREATE,
                StandardWatchEventKinds.ENTRY_DELETE, StandardWatchEventKinds.ENTRY_MODIF
Y);
            Map<String, BufferedReader> readers = new HashMap<>();
            Map<String, Long> lastLengths = new HashMap<>();
            int count = 0;
            while (true) {
                WatchKey watchKey = watchService.take();
                for (WatchEvent<?> event : watchKey.pollEvents()) {
                    if (event.kind() == StandardWatchEventKinds.ENTRY_MODIFY) {
                        Path filePath = dirPath.resolve((Path) event.context());
                        if (filePath.toString().endsWith(".csv")) {
                            File file = filePath.toFile();
                            long length = file.length();
                            if (!readers.containsKey(filePath.toString())) {
                                RandomAccessFile raf = new RandomAccessFile(file, "r");
                                readers.put(filePath.toString(),
                                    new BufferedReader(new InputStreamReader(new FileInpu
tStream(raf.getFD()), "GBK")));
                                lastLengths.put(filePath.toString(), 0L);
                            }
                            BufferedReader reader = readers.get(filePath.toString());
                            long lastLength = lastLengths.get(filePath.toString());
                            if (length > lastLength) {
                                RandomAccessFile raf = new RandomAccessFile(file, "r");
                                raf.seek(lastLength);
                                reader = new BufferedReader(new InputStreamReader(new Fil
eInputStream(raf.getFD()), "GBK"));
                                String line;
                                while ((line = reader.readLine()) != null) {
                                    log.info("New data in " + filePath + ": " + line);
                                    count++;
                                }
                                lastLengths.put(filePath.toString(), length);
                            }
                        }
                    }
```

```
                }
                if (!watchKey.reset()) {
                    break;
                }
                log.info("New data count  "  + count);
            }
            for (BufferedReader reader : readers.values()) {
                reader.close();
            }
        } catch (Exception e) {
            log.info("watch response error",e);
        }
    }
}
package com.chengyu.amlias.common.schema;
import io.netty.buffer.ByteBuf;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import java.util.Map;
import java.util.Map.Entry;
public abstract class MapSchema<K, V> extends BasicField<Entry<K, V>> {
    public final Logger log = LoggerFactory.getLogger(this.getClass().getSimpleName());
    public final Schema<K> keySchema;
    public final int lengthUnit;
    public final IntTool intTool;
    public final Map<K, Schema> valueSchema;

    public MapSchema(Schema<K> keySchema, int lengthUnit) {
        this.keySchema = keySchema;
        this.lengthUnit = lengthUnit;
        this.intTool = IntTool.getInstance(lengthUnit);
        PrepareLoadStrategy<K> loadStrategy = new PrepareLoadStrategy<>();
        addSchemas(loadStrategy);
        this.valueSchema = loadStrategy.build();
    }

    protected abstract void addSchemas(PrepareLoadStrategy<K> schemaRegistry);

    @Override
    public KeyValuePair<K, V> readFrom(ByteBuf input) {
        K key = keySchema.readFrom(input);
        KeyValuePair<K, V> result = new KeyValuePair<>(key);

        int length = intTool.read(input);
        if (length > 0) {
            int writerIndex = input.writerIndex();
            input.writerIndex(input.readerIndex() + length);

            Schema<V> schema = valueSchema.get(key);
            if (schema != null) {
```

```
                V value = schema.readFrom(input, length);
                result.setValue(value);
            } else {
                byte[] bytes = new byte[length];
                input.readBytes(bytes);
                result.setValue((V) bytes);
            }
            input.writerIndex(writerIndex);

        } else if (length < 0) {
            Schema<V> schema = valueSchema.get(key);
            if (schema != null) {
                V value = schema.readFrom(input);
                result.setValue(value);
            } else {
                byte[] bytes = new byte[input.readableBytes()];
                input.readBytes(bytes);
                result.setValue((V) bytes);
            }
        }
        return result;
    }

    @Override
    public void writeTo(ByteBuf output, Entry<K, V> entry) {
        if (entry == null)
            return;
        K key = entry.getKey();
        keySchema.writeTo(output, key);

        Schema schema = valueSchema.get(key);
        if (schema != null) {
            int begin = output.writerIndex();
            intTool.write(output, 0);

            Object value = entry.getValue();
            if (value != null) {
                schema.writeTo(output, value);
                int length = output.writerIndex() - begin - lengthUnit;
                intTool.set(output, begin, length);
            }
        } else {
            log.warn("未注册的信息:ID[{}], Value[{}]", key, entry.getValue());
        }
    }
}
package com.chengyu.amlias.service.impl;
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;
import com.chengyu.amlias.mapper.InvestigationRecordMapper;
import com.chengyu.amlias.domain.InvestigationRecord;
```

```
import com.chengyu.amlias.service.IInvestigationRecordService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;
@Service
public class InvestigationRecordServiceImpl extends ServiceImpl<InvestigationRecordMapper,
InvestigationRecord> implements IInvestigationRecordService{
    @Autowired
    private InvestigationRecordMapper investigationRecordMapper;

    @Override
    public InvestigationRecord getInvestigationRecordByRecordId(Integer recordId){
        return investigationRecordMapper.selectInvestigationRecordByRecordId(recordId);
    }

    @Override
    public List<InvestigationRecord> listInvestigationRecord(InvestigationRecord investig
ationRecord){
        return investigationRecordMapper.selectInvestigationRecordList(investigationRecor
d);
    }

    @Transactional
    @Override
    public int saveInvestigationRecord(InvestigationRecord investigationRecord){
        return investigationRecordMapper.insertInvestigationRecord(investigationRecord);
    }

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

    @Override
    @Transactional
    public int delInvestigationRecordByRecordIds(Integer[] recordIds){
        return investigationRecordMapper.deleteInvestigationRecordByRecordIds(recordIds);
    }

    private void investigationRecordData(List<InvestigationRecordDto> list) {
        List<String> list = list.stream().map(InvestigationRecordDto::getRecordId)
                .collect(Collectors.toList());
        list.stream().forEach(investigationRecord -> {
            InvestigationRecord investigationRecordEntry = entryInvestigationRecordMap.ge
t(investigationRecord.getRecordId);
```

```
            if (investigationRecordEntry != null) {
                investigationRecord.setInvestigationDate(InvestigationRecordTrans(investi
gationRecordEntry.getInvestigationDate));
                investigationRecord.setFindings(InvestigationRecordTrans(investigationRec
ordEntry.getFindings));
                investigationRecord.setAlertId(InvestigationRecordTrans(investigationReco
rdEntry.getAlertId));
            }
        });
    }

    @Override
    @Transactional
    public int updateInvestigationRecord(InvestigationRecord investigationRecord){
        return investigationRecordMapper.updateInvestigationRecord(investigationRecord);
    }

    @Override
    @Transactional
    public int delInvestigationRecordByRecordId(Integer recordId){
        return investigationRecordMapper.deleteInvestigationRecordByRecordId(recordId);
    }
}
package com.chengyu.amlias.mapper;
import java.util.List;
import com.chengyu.amlias.domain.MLModel;
import com.baomidou.mybatisplus.core.mapper.BaseMapper;
/**
 * 机器学习模型 Dao 接口
 */
public interface MLModelMapper extends BaseMapper<MLModel>{
    /**
     * 新增机器学习模型
     *
     * @param mLModel 机器学习模型
     * @return
     */
    int insertMLModel(MLModel mLModel);

    /**
     * 修改机器学习模型
     *
     * @param mLModel 机器学习模型
     * @return
     */
    int updateMLModel(MLModel mLModel);

    /**
     * 查询机器学习模型列表
     *
```

```
     * @param modelId 机器学习模型主键
     * @return
     */
    MLModel selectMLModelByModelId(Integer modelId);

    /**
     * 根据条件查询机器学习模型
     */
    List<MLModelVo> selectByCondition(MLModelQueryDto queryMLModelDto);

    /**
     * 删除机器学习模型
     * @param modelId 机器学习模型主键
     * @return
     */
    int deleteMLModelByModelId(Integer modelId);

    /**
     * 批量删除机器学习模型
     * @param modelIds  ID集合
     * @return
     */
    int deleteMLModelByModelIds(Integer[] modelIds);
}

package com.chengyu.amlias.domain;
import java.math.BigDecimal;
import java.util.Date;
import com.fasterxml.jackson.annotation.JsonFormat;
import lombok.Data;
import com.chengyu.amlias.common.annotation.Excel;
import com.baomidou.mybatisplus.annotation.TableName;
import com.chengyu.amlias.common.core.domain.BaseEntity;
/**
 * 账户信息实体类
 */
@TableName("amlias_account")
@Data
public class Account extends BaseEntity {
    private static final long serialVersionUID = -3013597246319531187LL;
    // 账户ID
    private Integer accountId;
    // 账户状态
    private String accountStatus;
    // 所属分行ID
    private Integer branchId;
    // 账户余额
    private BigDecimal balance;
    // 客户ID
    private Integer customerId;
```

```
    // 创建时间
    private Date createdAt;
    // 销户日期
    private String closeDate;
    // 货币类型
    private String currency;
    // 账户类型
    private String accountType;
    // 监控原因
    private String monitorReason;
    // 是否被监控
    private Integer isMonitored;
    // 账户号码
    private String accountNumber;
    // 更新时间
    private Date updatedAt;
    // 开户日期
    private String openDate;
}
package com.chengyu.amlias.controller;
import com.chengyu.amlias.common.constants.Constants;
import com.chengyu.amlias.common.entity.*;
import com.chengyu.amlias.common.entity.model.JsonResponse;
import com.chengyu.amlias.common.entity.model.TablePageForm;
import com.chengyu.amlias.protocol.JobExecuteKind;
import org.apache.commons.lang3.StringUtils;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.http.HttpStatus;
import com.chengyu.amlias.common.entity.model.TableResponse;
import com.chengyu.amlias.common.entity.vo.HeraActionVo;
import com.chengyu.amlias.common.entity.vo.HeraGroupVo;
import com.chengyu.amlias.common.entity.vo.HeraJobVo;
import com.chengyu.amlias.common.entity.vo.PageHelperTimeRange;
import com.chengyu.amlias.common.enums.*;
import com.chengyu.amlias.common.exception.NoPermissionException;
import com.chengyu.amlias.common.service.*;
import com.chengyu.amlias.common.util.ActionUtil;
import com.chengyu.amlias.logs.MonitorLog;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.*;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.TimeoutException;
import java.util.stream.Collectors;
/**
 * 调度中心管理
 */
@Controller
```

```
@RequestMapping("/scheduleCenter")
public class ScheduleCenterController extends BaseHeraController {

    private final HeraJobService heraJobService;
    private final HeraJobActionService heraJobActionService;
    private final HeraGroupService heraGroupService;
    private final HeraJobHistoryService heraJobHistoryService;
    private final HeraJobMonitorService heraJobMonitorService;
    private final HeraUserService heraUserService;
    private final HeraPermissionService heraPermissionService;
    private final WorkClient workClient;
    private final HeraHostGroupService heraHostGroupService;
    private final HeraAreaService heraAreaService;
    private final HeraSsoService heraSsoService;

    private void encryption(Map<String, String> config) {
        Optional.ofNullable(config)
                .ifPresent(cxf -> cxf.entrySet()
                        .stream()
                        .filter(pair -> pair.getKey().toLowerCase().contains(Constants.PA
SSWORD_WORD))
                        .forEach(entry -> entry.setValue("******")));
    }

    /**
     * 组下搜索任务
     *
     * @param groupId  groupId
     * @param status   all:全部;
     * @param pageForm layui table 分页参数
     * @return 结果
     */
    @RequestMapping(value = "/getGroupTask", method = RequestMethod.GET)
    @ResponseBody
    @ApiOperation(value = "某组下搜索任务")
    public TableResponse getGroupTask(@ApiParam(value = "组 Id", required = true) String g
roupId,
                                      @ApiParam(value = "任务状态") String status,
                                      @ApiParam(value = "日期", required = true) String d
t,
                                      @ApiParam(value = "分页参数", required = true) Tabl
ePageForm pageForm) {

        List<HeraGroup> group = heraGroupService.findDownStreamGroup(StringUtil.getGroupI
d(groupId));

        Set<Integer> groupSet = group.stream().map(HeraGroup::getId).collect(Collectors.t
oSet());
        List<HeraJob> jobList = heraJobService.getAll();
        Set<Integer> jobIdSet = jobList.stream().filter(job -> groupSet.contains(job.getG
```

```
roupId())).map(HeraJob::getId).collect(Collectors.toSet());
        SimpleDateFormat format = new SimpleDateFormat("yyMMdd");
        Calendar calendar = Calendar.getInstance();
        String startDate;
        Date start;
        if (StringUtils.isBlank(dt)) {
            start = new Date();
        } else {
            try {
                start = format.parse(dt);
            } catch (ParseException e) {
                start = new Date();
            }
        }

        calendar.setTime(start);
        startDate = ActionUtil.getFormatterDate("yyyyMMdd", calendar.getTime());
        calendar.add(Calendar.DAY_OF_YEAR, +1);
        String endDate = ActionUtil.getFormatterDate("yyyyMMdd", calendar.getTime());
        List<GroupTaskVo> taskVos = heraJobActionService.findByJobIds(new ArrayList<>(job
IdSet), startDate, endDate, pageForm, status);
        return new TableResponse(pageForm.getCount(), 0, taskVos);
    }

    public ScheduleCenterController(HeraJobMonitorService heraJobMonitorService,
        @Qualifier("heraJobMemoryService") HeraJobService heraJobService, HeraJobActionSe
rvice heraJobActionService,
        @Qualifier("heraGroupMemoryService") HeraGroupService heraGroupService, HeraJobHi
storyService heraJobHistoryService,
        HeraUserService heraUserService, HeraPermissionService heraPermissionService,
        WorkClient workClient, HeraHostGroupService heraHostGroupService, HeraAreaService
heraAreaService, HeraSsoService heraSsoService) {
        this.heraJobMonitorService = heraJobMonitorService;
        this.heraJobService = heraJobService;
        this.heraJobActionService = heraJobActionService;
        this.heraGroupService = heraGroupService;
        this.heraJobHistoryService = heraJobHistoryService;
        this.heraUserService = heraUserService;
        this.heraPermissionService = heraPermissionService;
        this.workClient = workClient;
        this.heraHostGroupService = heraHostGroupService;
        this.heraAreaService = heraAreaService;
        this.heraSsoService = heraSsoService;
    }

    @RequestMapping(method = RequestMethod.GET)
    @ApiOperation(value = "调度中心页面跳转")
    public String login() {
        return "scheduleCenter/scheduleCenter.index";
    }
```

```
    @RequestMapping(value = "/init", method = RequestMethod.POST)
    @ResponseBody
    @ApiOperation(value = "获取任务树,包含我的任务和全部任务两个集合")
    public JsonResponse initJobTree() {
        return new JsonResponse(true, Optional.of(heraJobService.buildJobTree(getOwner(),
Integer.parseInt(getSsoId()))).get());
    }

    @RequestMapping(value = "/getJobMessage", method = RequestMethod.GET)
    @ResponseBody
    @ApiOperation(value = "获取任务信息")
    public JsonResponse getJobMessage(@ApiParam(value = "任务 ID", required = true) Intege
r jobId) {
        HeraJob job = heraJobService.findById(jobId);
        HeraJobVo heraJobVo = BeanConvertUtils.convert(job);
        heraJobVo.setInheritConfig(getInheritConfig(job.getGroupId()));
        HeraJobMonitor monitor = heraJobMonitorService.findByJobId(jobId);
        StringBuilder focusUsers = new StringBuilder("[ ");
        Optional.ofNullable(monitor).ifPresent(m -> {
            if (StringUtils.isNotBlank(m.getUserIds())) {
                String ssoId = getSsoId();
                Arrays.stream(monitor.getUserIds().split(Constants.COMMA)).filter(StringU
tils::isNotBlank).distinct().forEach(id -> {
                    if (ssoId.equals(id)) {
                        heraJobVo.setFocus(true);
                        focusUsers.append(Constants.BLANK_SPACE).append(getSsoName());
                    } else {
                        Optional.ofNullable(heraSsoService.findSsoById(Integer.parseInt(i
d)))
                                .ifPresent(sso -> focusUsers.append(Constants.BLANK_SPACE).
append(sso.getName()));
                    }
                });
            }
        });
        focusUsers.append(" ]");
        Optional.ofNullable(heraHostGroupService.findById(job.getHostGroupId()))
                .ifPresent(group -> heraJobVo.setHostGroupName(group.getName()));
        heraJobVo.setUIdS(getuIds(jobId, RunAuthType.JOB));
        heraJobVo.setFocusUser(focusUsers.toString());
        heraJobVo.setAlarmLevel(AlarmLevel.getName(job.getOffset()));
        heraJobVo.setCycle(CycleEnum.parse(job.getCycle()).getDesc());
        configDecry(heraJobVo.getConfigs());
        configDecry(heraJobVo.getInheritConfig());
        //如果无权限，进行变量加密
        if (heraJobVo.getConfigs().keySet().stream().anyMatch(key -> key.toLowerCase().co
ntains(Constants.PASSWORD_WORD))
                || heraJobVo.getInheritConfig().keySet().stream().anyMatch(key -> key.toL
owerCase().contains(Constants.PASSWORD_WORD))) {
```

```
            try {
                checkPermission(jobId, RunAuthType.JOB);
            } catch (NoPermissionException e) {
                encryption(heraJobVo.getConfigs());
                encryption(heraJobVo.getInheritConfig());
            }
        }
        return new JsonResponse(true, heraJobVo);
    }


    @RunAuth(typeIndex = 1)
    @GetMapping("/checkPermission")
    @ResponseBody
    @ApiOperation(value = "权限检测接口")
    public JsonResponse doAspectAuth(@ApiParam(value = "任务 ID", required = true) Integer
jobId
            , @ApiParam(value = "检测类型", required = true) RunAuthType type) {
        return new JsonResponse(true, true);
    }


    private void configDecry(Map<String, String> config) {
        Optional.ofNullable(config)
                .ifPresent(cxf -> cxf.entrySet()
                        .stream()
                        .filter(pair -> pair.getKey().toLowerCase().contains(Constants.SE
CRET_PREFIX))
                        .forEach(entry -> entry.setValue(PasswordUtils.aesDecrypt(entry.g
etValue()))));
    }


    @RequestMapping(value = "/getGroupMessage", method = RequestMethod.GET)
    @ResponseBody
    @ApiOperation(value = "获取组信息")
    public JsonResponse getGroupMessage(@ApiParam(value = "组 ID", required = true) String
groupId) {
        Integer id = StringUtil.getGroupId(groupId);
        HeraGroup group = heraGroupService.findById(id);
        HeraGroupVo groupVo = BeanConvertUtils.convert(group);
        groupVo.setInheritConfig(getInheritConfig(groupVo.getParent()));
        groupVo.setUIdS(getuIds(id, RunAuthType.GROUP));
        configDecry(groupVo.getConfigs());
        configDecry(groupVo.getInheritConfig());
        if (groupVo.getConfigs().keySet().stream().anyMatch(key -> key.toLowerCase().cont
ains(Constants.PASSWORD_WORD))
                || groupVo.getInheritConfig().keySet().stream().anyMatch(key -> key.toLow
erCase().contains(Constants.PASSWORD_WORD))) {
            try {
```

```
                checkPermission(id, RunAuthType.GROUP);
            } catch (NoPermissionException e) {
                encryption(groupVo.getConfigs());
                encryption(groupVo.getInheritConfig());
            }
        }
        return new JsonResponse(true, groupVo);
    }


    @RequestMapping(value = "/getJobOperator", method = RequestMethod.GET)
    @ResponseBody
    @RunAuth(typeIndex = 1)
    @ApiOperation("获取任务管理员，admin 表示目前有权限操作的用户")
    public JsonResponse getJobOperator(@ApiParam(value = "任务/组 ID", required = true) St
ring jobId,
                                       @ApiParam(value = "任务类型", required = true) Run
AuthType type) {
        Integer groupId = StringUtil.getGroupId(jobId);
        List<HeraPermission> permissions = heraPermissionService.findByTargetId(groupId,
type.getName(), 1);
        List<HeraUser> all = heraUserService.findAllName();
        if (all == null || permissions == null) {
            return new JsonResponse(false, "发生错误，请联系管理员");
        }
        Map<String, Object> res = new HashMap<>(2);
        res.put("allUser", all);
        res.put("admin", permissions);
        return new JsonResponse(true, "查询成功", res);
    }

    @RequestMapping(value = "/getJobVersion", method = RequestMethod.GET)
    @ResponseBody
    @ApiOperation("获取任务的所有版本")
    public JsonResponse getJobVersion(@ApiParam(value = "任务 ID", required = true) Long j
obId) {
        return new JsonResponse(true, heraJobActionService.getActionVersionByJobId(jobId)
                .stream()
                .map(id -> HeraActionVo.builder().id(id).build())
                .collect(Collectors.toList()));
    }


    @RequestMapping(value = "/addJob", method = RequestMethod.POST)
    @ResponseBody
    @ApiOperation("新增任务")
    @RunAuth(authType = RunAuthType.GROUP, idIndex = 1)
    public JsonResponse addJob(@ApiParam(value = "任务信息", required = true) HeraJob her
aJob,
                               @ApiParam(value = "所在组目录", required = true) String pa
```

```
rentId) {
        heraJob.setGroupId(StringUtil.getGroupId(parentId));
        heraJob.setHostGroupId(HeraGlobalEnv.defaultWorkerGroup);
        heraJob.setOwner(getOwner());
        heraJob.setScheduleType(JobScheduleTypeEnum.Independent.getType());
        int insert = heraJobService.insert(heraJob);
        if (insert > 0) {
            addJobRecord(heraJob);
            return new JsonResponse(true, String.valueOf(heraJob.getId()));
        } else {
            return new JsonResponse(false, "新增失败");
        }
    }

    @PostMapping(value = "/copyJob")
    @ResponseBody
    @RunAuth(authType = RunAuthType.JOB)
    @ApiOperation("复制任务接口")
    public JsonResponse copyJob(
            @ApiParam(value = "复制的任务 ID", required = true) int jobId) {
        HeraJob copyJob = heraJobService.findById(jobId);
        copyJob.setName(copyJob.getName() + "_copy");
        copyJob.setOwner(getOwner());
        copyJob.setScheduleType(JobScheduleTypeEnum.Independent.getType());
        copyJob.setId(0);
        copyJob.setIsValid(0);
        int insert = heraJobService.insert(copyJob);
        if (insert > 0) {
            addJobRecord(copyJob);
            return new JsonResponse(true, String.valueOf(copyJob.getId()));
        } else {
            return new JsonResponse(false, "新增失败");
        }
    }


    private void addJobRecord(HeraJob heraJob) {
        String ssoName = getSsoName();
        String ownerId = getOwnerId();
        MonitorLog.info("{}[{}]【添加】任务{}成功", heraJob.getOwner(), ssoName, heraJob.
getId());
        updateMonitor(heraJob.getId());
        doAsync(() -> addJobRecord(heraJob.getId(), heraJob.getName(), RecordTypeEnum.Add,
ssoName, ownerId));
    }

    @RequestMapping(value = "/addMonitor", method = RequestMethod.POST)
    @ResponseBody
    @ApiOperation("任务添加监控")
    public JsonResponse updateMonitor(@ApiParam(value = "任务 ID", required = true) Intege
```

```
r id) {
        String ssoId = getSsoId();
        if (Constants.DEFAULT_ID.equals(ssoId)) {
            return new JsonResponse(false, "组账户不支持监控任务");
        }
        boolean res = heraJobMonitorService.addMonitor(ssoId, id);
        if (res) {
            MonitorLog.info("{}【关注】任务{}成功", getSsoName(), id);
            return new JsonResponse(true, "关注成功");
        } else {
            return new JsonResponse(false, "系统异常，请联系管理员");
        }
    }

    @RequestMapping(value = "/addGroup", method = RequestMethod.POST)
    @ResponseBody
    @RunAuth(authType = RunAuthType.GROUP, idIndex = 1)
    @ApiOperation("添加组")
    public JsonResponse addJob(@ApiParam(value = "组信息", required = true) HeraGroup her
aGroup,
                               @ApiParam(value = "所在组 id", required = true) String pare
ntId) {
        heraGroup.setParent(StringUtil.getGroupId(parentId));
        heraGroup.setOwner(getOwner());
        heraGroup.setExisted(1);
        int insert = heraGroupService.insert(heraGroup);
        if (insert > 0) {
            MonitorLog.info("{}【添加】组{}成功", getSsoName(), heraGroup.getId());
            return new JsonResponse(true, Constants.GROUP_PREFIX + heraGroup.getId());
        } else {
            return new JsonResponse(false, String.valueOf(-1));

        }
    }

    /**
     * 获取任务历史版本
     *
     * @param pageHelper
     * @return
     */
    @RequestMapping(value = "/getJobHistory", method = RequestMethod.GET)
    @ResponseBody
    @ApiOperation("任务历史记录")
    public JsonResponse getJobHistory(@ApiParam(value = "分页", required = true) PageHelp
erTimeRange pageHelper) {
        return new JsonResponse(true, heraJobHistoryService.findLogByPage(pageHelper));
    }

    @RequestMapping(value = "/getHostGroupIds", method = RequestMethod.GET)
```

```
    @ResponseBody
    @ApiOperation("获取机器组 Id")
    public JsonResponse getHostGroupIds() {
        return new JsonResponse(true, heraHostGroupService.getAll());
    }

    @RequestMapping(value = "getLog", method = RequestMethod.GET)
    @ResponseBody
    @RunAuth()
    @ApiOperation("获取任务日志接口")
    public JsonResponse getJobLog(@ApiParam(value = "任务 ID", required = true) Integer id)
{
        return new JsonResponse(true, heraJobHistoryService.findLogById(id));
    }

    @RequestMapping(value = "jobInstLog", method = RequestMethod.GET)
    @ResponseBody
    @RunAuth()
    @ApiOperation("获取任务日志接口")
    public String jobInstLog(@ApiParam(value = "任务 ID", required = true) Integer id,Inte
ger hisId) {
        return  heraJobHistoryService.findLogById(hisId).getLog();
    }


    @RequestMapping(value = "/status/{jobId}", method = RequestMethod.GET)
    @ResponseBody
    @UnCheckLogin
    @ApiOperation("开放接口,查询任务状态")
    public JsonResponse getStatus(@PathVariable("jobId") @ApiParam(value = "任务 ID", requ
ired = true) Integer jobId
            , @RequestParam("time") @ApiParam(value = "时间戳，只查询该时间戳之后的记录",
required = true) long time) {
        HeraJobHistory history = heraJobHistoryService.findNewest(jobId);
        if (history == null) {
            return new JsonResponse(false, "无执行记录");
        }
        //此时可能正在创建动态集群 或者发送 netty 消息的路上
        if (history.getStartTime() == null && history.getGmtCreate().getTime() >= time) {
            return new JsonResponse(true, StatusEnum.RUNNING.toString());
        }

        if (history.getStartTime() != null && history.getStartTime().getTime() < time) {
            return new JsonResponse(false, "无执行记录");
        }
        return new JsonResponse(true, Optional.ofNullable(history.getStatus()).orElse(Sta
tusEnum.RUNNING.toString()));
    }
}
package com.chengyu.amlias.controller;
```

```
import java.util.List;
import javax.servlet.http.HttpServletResponse;
import com.chengyu.amlias.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.amlias.common.annotation.Log;
import com.chengyu.amlias.common.core.controller.BaseController;
import com.chengyu.amlias.common.utils.poi.ExcelUtil;
import com.chengyu.amlias.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.amlias.common.enums.BusinessType;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import com.chengyu.amlias.domain.Customer;
import com.chengyu.amlias.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.setEmail(CustomerConvUtil(i.getEmail));
            i.setIndustry(CustomerConvUtil(i.getIndustry));
            i.setOccupation(CustomerConvUtil(i.getOccupation));
        });
        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.setEmail(CustomerExportFormat(i.getEmail));
            i.setIndustry(CustomerExportFormat(i.getIndustry));
            i.setOccupation(CustomerExportFormat(i.getOccupation));
        });
        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));
    }
}
<?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.amlias.mapper.MLModelMapper">
    <resultMap type="MLModel" id="MLModelResult">
        <result property="modelId" column="model_ID"    />
        <result property="updatedAt" column="updated_at"    />
        <result property="modelType" column="model_type"    />
        <result property="trainingData" column="training_data"    />
        <result property="modelName" column="model_name"    />
        <result property="createdAt" column="created_at"    />
        <result property="accuracy" column="accuracy"    />
    </resultMap>
    <sql id="selectMLModelVo">
        select model_ID, updated_at, model_type, training_data, model_name, created_at, a
ccuracy
        from amlias_mlmodel
    </sql>
```

```
    <select id="selectMLModel1List" parameterType="MLModel1" resultMap="MLModel1Result">
        <include refid="selectMLModel1Vo"/>
        <where>
            <if test="model1Id != null ">
            and model1_ID = #{model1Id}
            </if>
            <if test="updatedAt != null ">
            and updated_at = #{updatedAt}
            </if>
            <if test="model1Type != null and model1Type != ''">
            and model1_type = #{model1Type}
            </if>
            <if test="trainingData != null and trainingData != ''">
            and training_data = #{trainingData}
            </if>
            <if test="model1Name != null and model1Name != ''">
            and model1_name = #{model1Name}
            </if>
            <if test="createdAt != null ">
            and created_at = #{createdAt}
            </if>
            <if test="accuracy != null ">
            and accuracy = #{accuracy}
            </if>
        </where>
    </select>
    <delete id="deleteMLModel1ByModel1Id" parameterType="Integer">
        delete from am1ias_m1model1 where model1_ID = #{model1Id}
    </delete>

    <insert id="insertMLModel1" parameterType="MLModel1" useGeneratedKeys="true" keyPropert
y="model1Id">
        insert into am1ias_m1model1
        <trim prefix="(" suffix=")" suffixOverrides=",">
            <if test="updatedAt != null">updated_at,</if>
            <if test="model1Type != null and model1Type != ''">model1_type,</if>
            <if test="trainingData != null and trainingData != ''">training_data,</if>
            <if test="model1Name != null and model1Name != ''">model1_name,</if>
            <if test="createdAt != null">created_at,</if>
            <if test="accuracy != null">accuracy,</if>
         </trim>
        <trim prefix="values (" suffix=")" suffixOverrides=",">
            <if test="updatedAt != null">#{updatedAt},</if>
            <if test="model1Type != null and model1Type != ''">#{model1Type},</if>
            <if test="trainingData != null and trainingData != ''">#{trainingData},</if>
            <if test="model1Name != null and model1Name != ''">#{model1Name},</if>
            <if test="createdAt != null">#{createdAt},</if>
            <if test="accuracy != null">#{accuracy},</if>
         </trim>
    </insert>
```

```
    <delete id="deleteMLModelByModelIds" parameterType="String">
        delete from am1ias_m1model where model_ID in
        <foreach item="model1Id" collection="array" open="(" separator="," close=")">
            #{model1Id}
        </foreach>
    </delete>
</mapper>
<?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.am1ias.mapper.InvestigationRecordMapper">
    <resultMap type="InvestigationRecord" id="InvestigationRecordResult">
        <result property="recordId" column="record_ID"    />
        <result property="recommendation" column="recommendation"    />
        <result property="investigationDate" column="investigation_date"    />
        <result property="investigator" column="investigator"    />
        <result property="alertId" column="alert_ID"    />
        <result property="conclusion" column="conclusion"    />
        <result property="createdAt" column="created_at"    />
        <result property="updatedAt" column="updated_at"    />
        <result property="findings" column="findings"    />
    </resultMap>
    <sql id="selectInvestigationRecordVo">
        select record_ID, recommendation, investigation_date, investigator, alert_ID, con
clusion, created_at, updated_at, findings
        from am1ias_investigationrecord
    </sql>
    <select id="selectInvestigationRecordList" parameterType="InvestigationRecord" result
Map="InvestigationRecordResult">
        <include refid="selectInvestigationRecordVo"/>
        <where>
            <if test="recordId != null ">
            and record_ID = #{recordId}
            </if>
            <if test="recommendation != null and recommendation != ''">
            and recommendation = #{recommendation}
            </if>
            <if test="investigationDate != null and investigationDate != ''">
            and investigation_date = #{investigationDate}
            </if>
            <if test="investigator != null and investigator != ''">
            and investigator = #{investigator}
            </if>
            <if test="alertId != null ">
            and alert_ID = #{alertId}
            </if>
            <if test="conclusion != null and conclusion != ''">
            and conclusion = #{conclusion}
            </if>
            <if test="createdAt != null ">
```

```
            and created_at = #{createdAt}
            </if>
            <if test="updatedAt != null ">
            and updated_at = #{updatedAt}
            </if>
            <if test="findings != null and findings != ''">
            and findings = #{findings}
            </if>
        </where>
    </select>
    <delete id="deleteInvestigationRecordByRecordId" parameterType="Integer">
        delete from amlias_investigationrecord where record_ID = #{recordId}
    </delete>

    <insert id="insertInvestigationRecord" parameterType="InvestigationRecord" useGenerat
edKeys="true" keyProperty="recordId">
        insert into amlias_investigationrecord
        <trim prefix="(" suffix=")" suffixOverrides=",">
            <if test="recommendation != null and recommendation != ''">recommendation,</i
f>
            <if test="investigationDate != null and investigationDate != ''">investigatio
n_date,</if>
            <if test="investigator != null and investigator != ''">investigator,</if>
            <if test="alertId != null">alert_ID,</if>
            <if test="conclusion != null and conclusion != ''">conclusion,</if>
            <if test="createdAt != null">created_at,</if>
            <if test="updatedAt != null">updated_at,</if>
            <if test="findings != null and findings != ''">findings,</if>
         </trim>
        <trim prefix="values (" suffix=")" suffixOverrides=",">
            <if test="recommendation != null and recommendation != ''">#{recommendation},
</if>
            <if test="investigationDate != null and investigationDate != ''">#{investigat
ionDate},</if>
            <if test="investigator != null and investigator != ''">#{investigator},</if>
            <if test="alertId != null">#{alertId},</if>
            <if test="conclusion != null and conclusion != ''">#{conclusion},</if>
            <if test="createdAt != null">#{createdAt},</if>
            <if test="updatedAt != null">#{updatedAt},</if>
            <if test="findings != null and findings != ''">#{findings},</if>
         </trim>
    </insert>
    <delete id="deleteInvestigationRecordByRecordIds" parameterType="String">
        delete from amlias_investigationrecord where record_ID in
        <foreach item="recordId" collection="array" open="(" separator="," close=")">
            #{recordId}
        </foreach>
    </delete>
</mapper>
package com.chengyu.amlias.common.processor;
```

```
import com.alibaba.fastjson.JSONObject;
import lombok.Builder;
import lombok.Data;
import java.util.Map;
import java.util.stream.Collectors;
@Builder
@Data
public class JobProcessor implements Processor {

    public static final String name = "JobProcessor";

    private String jobId;
    private Map<String, String> kvConfig;

    @Override
    public String getId() {
        return name;
    }

    @Override
    public String getConfig() {
        JSONObject jsonObject = new JSONObject();
        jsonObject.put("actionId", jobId);
        JSONObject kv = new JSONObject();
        if (kvConfig != null) {
            kvConfig.keySet()
                    .stream()
                    .filter(key -> key.startsWith("instance."))
                    .forEach(value -> kv.put(value, kvConfig.get(value)));
        }
        jsonObject.put("kvConfig", kv);
        return jsonObject.toString();
    }

    @Override
    public void parse(String config) {
        JSONObject jsonObject = JSONObject.parseObject(config);
        jobId = jsonObject.getString("actionId");
        kvConfig = jsonObject.getJSONObject("kvConfig").entrySet().stream()
                             .filter(ks -> jsonObject.getString(ks.toString()) != null)
                             .collect(Collectors.toMap(k -> k.toString(), v -> jsonObject.g
etString(v.toString()), (s, t) -> t));

    }
}
package com.chengyu.amlias.common.service.impl;
import com.chengyu.amlias.common.entity.HeraHostGroup;
import com.chengyu.amlias.common.entity.HeraHostRelation;
import com.chengyu.amlias.common.mapper.HeraHostGroupMapper;
import com.chengyu.amlias.common.service.HeraHostRelationService;
```

```
import com.chengyu.amlias.common.entity.vo.HeraHostGroupVo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.context.annotation.PropertySource;
import org.springframework.stereotype.Service;
import com.chengyu.amlias.common.service.HeraHostGroupService;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
@Service("heraHostGroupService")
public class HeraHostGroupServiceImpl implements HeraHostGroupService {


    @Autowired
    private HeraHostGroupMapper heraHostGroupMapper;
    @Autowired
    private HeraHostRelationService heraHostRelationService;

    @Override
    public int insert(HeraHostGroup heraHostGroup) {
        return heraHostGroupMapper.insert(heraHostGroup);
    }

    @Override
    public int delete(int id) {
        return heraHostGroupMapper.delete(id);
    }

    @Override
    public int update(HeraHostGroup heraHostGroup) {
        return heraHostGroupMapper.update(heraHostGroup);
    }

    @Override
    public List<HeraHostGroup> getAll() {
        return heraHostGroupMapper.getAll();
    }

    @Override
    public HeraHostGroup findById(int id) {
        HeraHostGroup group = HeraHostGroup.builder().id(id).build();
        return heraHostGroupMapper.findById(group);
    }

    @Override
    public Map<Integer, HeraHostGroupVo> getAllHostGroupInfo() {
        List<HeraHostGroup> groupList = this.getAll();
        Map<Integer, HeraHostGroupVo> hostGroupInfoMap = new HashMap<>(groupList.size());
        List<HeraHostRelation> relationList = heraHostRelationService.getAll();
```

```
        groupList.forEach(heraHostGroup -> {
            if(heraHostGroup.getEffective() == 1) {
                HeraHostGroupVo vo = HeraHostGroupVo.builder()
                        .id(String.valueOf(heraHostGroup.getId()))
                        .name(heraHostGroup.getName())
                        .nextPos(0)
                        .description(heraHostGroup.getDescription())
                        .build();
                List<String> hosts = new ArrayList<>();
                relationList.forEach(heraHostRelation -> {
                    if(heraHostRelation.getHostGroupId() == (heraHostGroup.getId())) {
                        hosts.add(heraHostRelation.getHost());
                    }
                });
                vo.setHosts(hosts);
                hostGroupInfoMap.put(heraHostGroup.getId(), vo);
            }
        });
        return hostGroupInfoMap;
    }
}
package com.chengyu.amlias.common.service.impl;
import com.chengyu.amlias.common.entity.HeraAdvice;
import com.chengyu.amlias.common.mapper.HeraAdviceMapper;
import com.chengyu.amlias.common.service.HeraAdviceService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;

@Service("heraAdviceService")
public class HeraHeraAdviceServiceImpl implements HeraAdviceService {


    @Autowired
    private HeraAdviceMapper heraAdviceMapper;

    @Override
    public boolean addAdvice(HeraAdvice heraAdvice) {
        Integer res = heraAdviceMapper.insert(heraAdvice);
        return res != null && res > 0;
    }

    @Override
    public List<HeraAdvice> getAll() {
        return heraAdviceMapper.getAll();
    }
}
package com.chengyu.amlias.common.service.impl;
import com.chengyu.amlias.common.entity.HeraHostRelation;
import com.chengyu.amlias.common.mapper.HeraHostRelationMapper;
```

```
import com.chengyu.amlias.common.service.HeraHostRelationService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;

@Service("heraHostRelationService")
public class HeraHostRelationServiceImpl implements HeraHostRelationService {

    @Autowired
    private HeraHostRelationMapper heraHostRelationMapper;

    @Override
    public int insert(HeraHostRelation heraHostRelation) {
        return heraHostRelationMapper.insert(heraHostRelation);
    }

    @Override
    public int delete(int id) {
        return heraHostRelationMapper.delete(id);
    }

    @Override
    public int update(HeraHostRelation heraHostRelation) {
        return heraHostRelationMapper.update(heraHostRelation);
    }

    @Override
    public List<HeraHostRelation> getAll() {
        return heraHostRelationMapper.getAll();
    }

    @Override
    public HeraHostRelation findById(HeraHostRelation heraHostRelation) {
        return heraHostRelationMapper.findById(heraHostRelation);
    }

    @Override
    public List<String> findPreemptionGroup(int id) {
        return heraHostRelationMapper.findPreemptionGroup(id);
    }
}
package com.chengyu.amlias.common.entity;
import com.chengyu.amlias.common.config.SkipColumn;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;
import java.util.Date;
```

```
@Builder
@Data
@AllArgsConstructor
@NoArgsConstructor
@ApiModel(value = "hera 任务组对象")
public class HeraGroup {

    @ApiModelProperty(value = "组 ID")
    private Integer id;
    @ApiModelProperty(value = "json 格式的配置")

    private String configs;
    @ApiModelProperty(value = "描述")

    private String description;
    @ApiModelProperty(value = "0 大目录，1 小目录")
    private Integer directory;

    @SkipColumn
    @ApiModelProperty(value = "创建时间")
    private Date gmtCreate;

    @SkipColumn
    private Date gmtModified;


    private String name;
    @ApiModelProperty(value = "所有人")
    private String owner;
    @ApiModelProperty(value = "父目录 id")
    private Integer parent;
    @ApiModelProperty(value = "目前无用")
    private String resources;
    @ApiModelProperty(value = "0 删除，1 存在")
    private Integer existed;

}

package com.chengyu.amlias.service.impl;
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;
import com.chengyu.amlias.mapper.AlertRecordMapper;
import com.chengyu.amlias.domain.AlertRecord;
import com.chengyu.amlias.service.IAlertRecordService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;
@Service
public class AlertRecordServiceImpl extends ServiceImpl<AlertRecordMapper, AlertRecord> i
mplements IAlertRecordService{
    @Autowired
```

```
    private AlertRecordMapper alertRecordMapper;

    @Override
    public AlertRecord getAlertRecordByAlertId(Integer alertId){
        return alertRecordMapper.selectAlertRecordByAlertId(alertId);
    }

    @Override
    public List<AlertRecord> listAlertRecord(AlertRecord alertRecord){
        return alertRecordMapper.selectAlertRecordList(alertRecord);
    }

    @Transactional
    @Override
    public int saveAlertRecord(AlertRecord alertRecord){
        return alertRecordMapper.insertAlertRecord(alertRecord);
    }

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

    @Override
    @Transactional
    public int delAlertRecordByAlertIds(Integer[] alertIds){
        return alertRecordMapper.deleteAlertRecordByAlertIds(alertIds);
    }

    private void alertRecordData(List<AlertRecordDto> list) {
        List<String> list = list.stream().map(AlertRecordDto::getAlertId)
                .collect(Collectors.toList());
        list.stream().forEach(alertRecord -> {
            AlertRecord alertRecordEntry = entryAlertRecordMap.get(alertRecord.getAlertId);

            if (alertRecordEntry != null) {
                alertRecord.setAlertTime(AlertRecordTrans(alertRecordEntry.getAlertTime));
                alertRecord.setAlertLevel(AlertRecordTrans(alertRecordEntry.getAlertLevel))
;
                alertRecord.setCreatedAt(AlertRecordTrans(alertRecordEntry.getCreatedAt));
                alertRecord.setAlertId(AlertRecordTrans(alertRecordEntry.getAlertId));
                alertRecord.setInvestigationStatus(AlertRecordTrans(alertRecordEntry.getI
nvestigationStatus));
            }
        });
```

```
    }

    @Override
    @Transactional
    public int updateAlertRecord(AlertRecord alertRecord){
        return alertRecordMapper.updateAlertRecord(alertRecord);
    }

    @Override
    @Transactional
    public int delAlertRecordByAlertId(Integer alertId){
        return alertRecordMapper.deleteAlertRecordByAlertId(alertId);
    }
}
package com.chengyu.amlias.utils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import java.io.UnsupportedEncodingException;
import java.net.URLDecoder;
import java.net.URLEncoder;
import java.util.Base64;
public class EncryptUtil {
    private static final Logger logger = LoggerFactory.getLogger(EncryptUtil.class);

    public static String encodeBase64(byte[] bytes){
        String encoded = Base64.getEncoder().encodeToString(bytes);
        return encoded;
    }

    public static byte[]  decodeBase64(String str){
        byte[] bytes = null;
        bytes = Base64.getDecoder().decode(str);
        return bytes;
    }

    public static String encodeUTF8StringBase64(String str){
        String encoded = null;
        try {
            encoded = Base64.getEncoder().encodeToString(str.getBytes("utf-8"));
        } catch (UnsupportedEncodingException e) {
            logger.warn("不支持的编码格式",e);
        }
        return encoded;

    }

    public static String  decodeUTF8StringBase64(String str){
        String decoded = null;
        byte[] bytes = Base64.getDecoder().decode(str);
        try {
```

```
            decoded = new String(bytes,"utf-8");
        }catch(UnsupportedEncodingException e){
            logger.warn("不支持的编码格式",e);
        }
        return decoded;
    }

    public static String encodeURL(String url) {
        String encoded = null;
        try {
            encoded =  URLEncoder.encode(url, "utf-8");
        } catch (UnsupportedEncodingException e) {
            logger.warn("URLEncode 失败", e);
        }
        return encoded;
    }


    public static String decodeURL(String url) {
        String decoded = null;
        try {
            decoded = URLDecoder.decode(url, "utf-8");
        } catch (UnsupportedEncodingException e) {
            logger.warn("URLDecode 失败", e);
        }
        return decoded;
    }

    public static void main(String [] args){
        String str = "abcd{'a':'b'}";
        String encoded = EncryptUtil.encodeUTF8StringBase64(str);
        String decoded = EncryptUtil.decodeUTF8StringBase64(encoded);
        System.out.println(str);
        System.out.println(encoded);
        System.out.println(decoded);

        String url = "== wo";
        String urlEncoded = EncryptUtil.encodeURL(url);
        String urlDecoded = EncryptUtil.decodeURL(urlEncoded);

        System.out.println(url);
        System.out.println(urlEncoded);
        System.out.println(urlDecoded);
    }
}
package com.chengyu.amlias.utils;
import org.apache.commons.lang.StringUtils;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
```

```
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.ZoneId;
import java.time.ZoneOffset;
import java.time.format.DateTimeFormatter;
import java.util.*;
/**
 * 日期处理
 *
 */
public class DateUtil {
    public static final String YYYY_MM_DD_HH_MM_SS = "yyyy-MM-dd HH:mm:ss";

    public static final String YYYYMMDD = "yyyyMMdd";

    public static final String HHmmss = "HHmmss";

    public static final String YYYYMM = "yyyyMM";

    private DateUtil() {
    }

    public static String toDateTime(LocalDateTime date) {
        return toDateTime(date, YYYY_MM_DD_HH_MM_SS);
    }

    public static String toDateTime(LocalDateTime dateTime, String pattern) {
        return dateTime.format(DateTimeFormatter.ofPattern(pattern, Locale.SIMPLIFIED_CHI
NESE));
    }


    public static String toDateText(LocalDate date, String pattern) {
        if (date == null || pattern == null) {
            return null;
        }
        return date.format(DateTimeFormatter.ofPattern(pattern, Locale.SIMPLIFIED_CHINESE))
;
    }

    /**
     * 从给定的 date，加上 hour 小时 求指定 date 时间后 hour 小时的时间
     *
     * @param date 指定的时间
     * @param hour 多少小时后
     * @return
     */
    public static Date addExtraHour(Date date, int hour) {
        Calendar cal = Calendar.getInstance();
        if (date != null) {
```

```
            cal.setTime(date);
        }
        cal.add(Calendar.HOUR_OF_DAY, hour);
        return cal.getTime();
    }

    /**
     * 从给定的 date，加上 increase 天
     *
     * @param date
     * @param increase
     * @return
     */
    public static Date increaseDay2Date(Date date, int increase) {
        Calendar cal = Calendar.getInstance();
        if (date != null) {
            cal.setTime(date);
        }
        cal.add(Calendar.DAY_OF_MONTH, increase);
        return cal.getTime();
    }

    /**
     * 从给定的 LocalDateTime，加上 increase 月
     *
     * @param date
     * @param increase
     * @return
     */
    public static LocalDateTime localDateTimeAddMonth(LocalDateTime date, int increase) {
        Date temp = Date.from(date.atZone(ZoneId.systemDefault()).toInstant());
        Calendar cal = Calendar.getInstance();
        if (temp != null) {
            cal.setTime(temp);
        }
        cal.add(Calendar.MONTH, increase);
        // LocalDateTime <-- Date
        return  LocalDateTime.ofInstant(cal.getTime().toInstant(), ZoneId.systemDefault());
    }


    /**
     * 把字符串日期默认转换为 yyyy-mm-dd 格式的 Data 对象
     *
     * @param strDate
     * @return
     */
    public static Date format(String strDate, String format) {
        Date d = null;
        if (null == strDate || "".equals(strDate))
```

```
            return null;
        else
            try {
                d = getFormatter(format).parse(strDate);
            } catch (ParseException pex) {
                return null;
            }
        return d;
    }

    /**
     * 获取一个简单的日期格式化对象
     *
     * @return 一个简单的日期格式化对象
     */
    private static SimpleDateFormat getFormatter(String parttern) {
        return new SimpleDateFormat(parttern);
    }

    /**
     * 获取 month 所在月的所有天
     *
     * @param month      要查询的日期（如果为 null 则默认为当前月）
     * @param dateFormat 返回日期的格式（如果为 null 则返回 yyyy-MM-dd 格式结果）
     * @return
     */
    public static List<String> getAllDaysOfMonthInString(Date month, DateFormat dateForma
t) {
        List<String> rs = new ArrayList<String>();
        DateFormat df = null;
        if (null == dateFormat) {
            df = new SimpleDateFormat("yyyy-MM-dd");
        }
        Calendar cad = Calendar.getInstance();
        if (null != month) {
            cad.setTime(month);
        }
        int day_month = cad.getActualMaximum(Calendar.DAY_OF_MONTH); // 获取当月天数
        for (int i = 0; i < day_month; i++) {
            cad.set(Calendar.DAY_OF_MONTH, i + 1);
            rs.add(df.format(cad.getTime()));

        }
        return rs;
    }

    /**
     * 获取 month 所在月的所有天
     *
     * @param month 要查询的日期（如果为 null 则默认为当前月）
```

```
     * @return 日期List
     */
    public static List<Date> getAllDaysOfMonth(Date month) {
        List<Date> rs = new ArrayList<Date>();
        Calendar cad = Calendar.getInstance();
        if (null != month) {
            cad.setTime(month);
        }
        int day_month = cad.getActualMaximum(Calendar.DAY_OF_MONTH); // 获取当月天数
        for (int i = 0; i < day_month; i++) {
            cad.set(Calendar.DAY_OF_MONTH, i + 1);
            rs.add(cad.getTime());

        }
        return rs;
    }

    /**
     * 获取指定日期区间所有天
     *
     * @param begin
     * @param end
     * @param dateFormat (如果为null 则返回 yyyy-MM-dd 格式的日期)
     * @return
     */
    public static List<String> getSpecifyDaysOfMonthInString(Date begin, Date end, DateFo
rmat dateFormat) {
        DateFormat df = null;
        if (null == dateFormat) {
            df = new SimpleDateFormat("yyyy-MM-dd");
        }
        List<String> rs = new ArrayList<String>();
        List<Date> tmplist = getSpecifyDaysOfMonth(begin, end);
        for (Date date : tmplist)
            rs.add(df.format(date));
        return rs;
    }

    /**
     * 获取指定日期区间所有天
     *
     * @param begin
     * @param end
     * @return
     */
    public static List<Date> getSpecifyDaysOfMonth(Date begin, Date end) {
        List<Date> rs = new ArrayList<Date>();
        Calendar cad = Calendar.getInstance();
        int day_month = -1;
        if (null == begin) {// 设置开始日期为指定日期
```

```
            // day_month = cad.getActualMaximum(Calendar.DAY_OF_MONTH); // 获取当月天数
            cad.set(Calendar.DAY_OF_MONTH, 1);// 设置开始日期为当前月的第一天
            begin = cad.getTime();
        }
        cad.setTime(begin);
        if (null == end) {// 如果结束日期为空 , 设置结束日期为下月的第一天
            day_month = cad.getActualMaximum(Calendar.DAY_OF_MONTH); // 获取当月天数
            cad.set(Calendar.DAY_OF_MONTH, day_month + 1);
            end = cad.getTime();
        }
        cad.set(Calendar.DAY_OF_MONTH, 1);// 设置开始日期为当前月的第一天
        Date tmp = begin;
        int i = 1;
        while (true) {
            cad.set(Calendar.DAY_OF_MONTH, i);
            i++;
            tmp = cad.getTime();
            if (tmp.before(end)) {
                rs.add(cad.getTime());
            } else {
                break;
            }
        }
        return rs;
    }

    /**
     * 获取当前日期
     *
     * @return 一个包含年月日的<code>Date</code>型日期
     */
    public static synchronized Date getCurrDate() {
        Calendar calendar = Calendar.getInstance();
        return calendar.getTime();
    }

    public static String format(Date date, String pattern) {
        SimpleDateFormat dateFormat = new SimpleDateFormat(pattern);
        return dateFormat.format(date);
    }

    /**
     * 获取当前完整时间,样式: yyyy－MM－dd hh:mm:ss
     *
     * @return 一个包含年月日时分秒的<code>String</code>型日期。yyyy-MM-dd hh:mm:ss
     */
    public static String getCurrDateTimeStr() {
        return format(getCurrDate(), YYYY_MM_DD_HH_MM_SS);
    }
```

```
    /**
     * 获得指定日期的前一天
     *
     * @param specifiedDay YYYY_MM_DD_HH_MM_SS 格式
     * @param formatStr    日期类型
     * @return
     */
    public static String getSpecifiedDayBefore(String specifiedDay, String formatStr) {//
可以用 new
        // Date().toLocalString()传递参数
        Calendar c = Calendar.getInstance();
        Date date = null;
        try {
            date = new SimpleDateFormat(YYYY_MM_DD_HH_MM_SS).parse(specifiedDay);
        } catch (ParseException e) {
            e.printStackTrace();
        }
        c.setTime(date);
        int day = c.get(Calendar.DATE);
        c.set(Calendar.DATE, day - 1);

        String dayBefore = new SimpleDateFormat(formatStr).format(c.getTime());
        return dayBefore;
    }
    /**
     * 获取当前月的第一天
     *
     * @return
     */
    public static final String getCurrentMonthFirstDay() {
        Calendar ca1 = Calendar.getInstance();
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        // 当前月的第一天
        ca1.set(GregorianCalendar.DAY_OF_MONTH, 1);
        Date beginTime = ca1.getTime();
        return sdf.format(beginTime);
    }

    /**
     * 获取昨天开始时间
     *
     * @return
     */
    public static final String getYesterdayStart() {
        Calendar ca1 = Calendar.getInstance();
        ca1.add(Calendar.DATE, -1);
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        return sdf.format(ca1.getTime());
    }
```

```
    public static final String getYesterdayEnd() {
        Calendar cal = Calendar.getInstance();
        cal.add(Calendar.DATE, -1);
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        return sdf.format(cal.getTime()) + " 23:59:59";
    }

    public static final String getCurrDayStart() {
        Calendar cal = Calendar.getInstance();
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        return sdf.format(cal.getTime());
    }
}
```