```
package com.cy.ftamlirtp.mapper;
import java.util.List;
import com.cy.ftamlirtp.domain.AlertRecord;
import com.baomidou.mybatisplus.core.mapper.BaseMapper;
/**
 * 预警记录 Dao 接口
 */
public interface AlertRecordMapper extends BaseMapper<AlertRecord>{
    /**
     * 新增预警记录
     *
     * @param alertRecord 预警记录
     * @return
     */
    int insertAlertRecord(AlertRecord alertRecord);

    /**
     * 修改预警记录
     *
     * @param alertRecord 预警记录
     * @return
     */
    int updateAlertRecord(AlertRecord alertRecord);

    /**
     * 查询预警记录列表
     *
     * @param alertId 预警记录主键
     * @return
     */
    AlertRecord selectAlertRecordByAlertId(Integer alertId);

    /**
     * 根据条件查询预警记录
     */
    List<AlertRecordVo> selectByCondition(AlertRecordQueryDto queryAlertRecordDto);

    /**
     * 删除预警记录
     * @param alertId 预警记录主键
     * @return
     */
    int deleteAlertRecordByAlertId(Integer alertId);

    /**
     * 批量删除预警记录
     * @param alertIds  ID 集合
     * @return
     */
    int deleteAlertRecordByAlertIds(Integer[] alertIds);
```

```
}

package com.cy.ftamlirtp.controller;
import java.util.List;
import javax.servlet.http.HttpServletResponse;
import com.cy.ftamlirtp.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.cy.ftamlirtp.common.annotation.Log;
import com.cy.ftamlirtp.common.core.controller.BaseController;
import com.cy.ftamlirtp.common.utils.poi.ExcelUtil;
import com.cy.ftamlirtp.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.cy.ftamlirtp.common.enums.BusinessType;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import com.cy.ftamlirtp.domain.Customer;
import com.cy.ftamlirtp.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.setLastReviewDate(CustomerConvUtil(i.getLastReviewDate));
            i.setOnboardDate(CustomerConvUtil(i.getOnboardDate));
            i.setStatus(CustomerConvUtil(i.getStatus));
            i.setDateOfBirth(CustomerConvUtil(i.getDateOfBirth));
            i.setRiskLevel(CustomerConvUtil(i.getRiskLevel));
            i.setUpdatedAt(CustomerConvUtil(i.getUpdatedAt));
            i.setNextReviewDate(CustomerConvUtil(i.getNextReviewDate));
            i.setCreatedAt(CustomerConvUtil(i.getCreatedAt));
            i.setPhoneNumber(CustomerConvUtil(i.getPhoneNumber));
            i.setName(CustomerConvUtil(i.getName));
            i.setNationality(CustomerConvUtil(i.getNationality));
        });
        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.setLastReviewDate(CustomerExportFormat(i.getLastReviewDate));
            i.setOnboardDate(CustomerExportFormat(i.getOnboardDate));
            i.setStatus(CustomerExportFormat(i.getStatus));
            i.setDateOfBirth(CustomerExportFormat(i.getDateOfBirth));
            i.setRiskLevel(CustomerExportFormat(i.getRiskLevel));
            i.setUpdatedAt(CustomerExportFormat(i.getUpdatedAt));
            i.setNextReviewDate(CustomerExportFormat(i.getNextReviewDate));
            i.setCreatedAt(CustomerExportFormat(i.getCreatedAt));
            i.setPhoneNumber(CustomerExportFormat(i.getPhoneNumber));
            i.setName(CustomerExportFormat(i.getName));
            i.setNationality(CustomerExportFormat(i.getNationality));
        });
        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.cy.ftamlirtp.mapper.ReportMapper">
    <resultMap type="Report" id="ReportResult">
        <result property="reportId" column="report_ID"    />
        <result property="reportType" column="report_type"    />
        <result property="reportDate" column="report_date"    />
        <result property="updatedAt" column="updated_at"    />
        <result property="alertId" column="alert_ID"    />
        <result property="status" column="status"    />
        <result property="createdAt" column="created_at"    />
        <result property="description" column="description"    />
        <result property="recipient" column="recipient"    />
    </resultMap>
    <sql id="selectReportVo">
        select report_ID, report_type, report_date, updated_at, alert_ID, status, created
_at, description, recipient
        from ftamlirtp_report
    </sql>
    <select id="selectReportList" parameterType="Report" resultMap="ReportResult">
        <include refid="selectReportVo"/>
        <where>
            <if test="reportId != null ">
            and report_ID = #{reportId}
            </if>
            <if test="reportType != null and reportType != ''">
            and report_type = #{reportType}
            </if>
            <if test="reportDate != null and reportDate != ''">
            and report_date = #{reportDate}
            </if>
            <if test="updatedAt != null ">
            and updated_at = #{updatedAt}
            </if>
            <if test="alertId != null ">
            and alert_ID = #{alertId}
            </if>
            <if test="status != null and status != ''">
            and status = #{status}
            </if>
            <if test="createdAt != null ">
            and created_at = #{createdAt}
            </if>
            <if test="description != null and description != ''">
            and description = #{description}
            </if>
            <if test="recipient != null and recipient != ''">
            and recipient = #{recipient}
            </if>
        </where>
    </select>
    <delete id="deleteReportByReportId" parameterType="Integer">
        delete from ftamlirtp_report where report_ID = #{reportId}
    </delete>

    <insert id="insertReport" parameterType="Report" useGeneratedKeys="true" keyProperty=
"reportId">
        insert into ftamlirtp_report
        <trim prefix="(" suffix=")" suffixOverrides=",">
            <if test="reportType != null and reportType != ''">report_type,</if>
            <if test="reportDate != null and reportDate != ''">report_date,</if>
            <if test="updatedAt != null">updated_at,</if>
            <if test="alertId != null">alert_ID,</if>
            <if test="status != null and status != ''">status,</if>
            <if test="createdAt != null">created_at,</if>
            <if test="description != null and description != ''">description,</if>
            <if test="recipient != null and recipient != ''">recipient,</if>
         </trim>
        <trim prefix="values (" suffix=")" suffixOverrides=",">
            <if test="reportType != null and reportType != ''">#{reportType},</if>
            <if test="reportDate != null and reportDate != ''">#{reportDate},</if>
            <if test="updatedAt != null">#{updatedAt},</if>
            <if test="alertId != null">#{alertId},</if>
            <if test="status != null and status != ''">#{status},</if>
            <if test="createdAt != null">#{createdAt},</if>
            <if test="description != null and description != ''">#{description},</if>
            <if test="recipient != null and recipient != ''">#{recipient},</if>
         </trim>
    </insert>
    <delete id="deleteReportByReportIds" parameterType="String">
        delete from ftamlirtp_report where report_ID in
        <foreach item="reportId" collection="array" open="(" separator="," close=")">
            #{reportId}
        </foreach>
    </delete>
</mapper>
package com.cy.ftamlirtp.adaptor.single;
import com.cy.ftamlirtp.core.common.algorithm.IAlgorithm;
import com.cy.ftamlirtp.core.common.datasource.CsvDataSource;
import com.cy.ftamlirtp.core.common.datasource.IDataSource;
import com.cy.ftamlirtp.core.common.table.Table;
import java.io.IOException;
import java.util.List;

public class SingleAdaptor {
  private IAlgorithm algorithm;

  public void setAlgorithm(IAlgorithm algorithm) {
    this.algorithm = algorithm;
  }

  public void run(String inputFilePath, String outputFilePath, final List<String> usedAtt
rs) throws IOException {
    IDataSource dataSource = new CsvDataSource(inputFilePath);
    if (usedAttrs != null) {
      dataSource.filter(usedAttrs.toArray(new String[]{}));
    }
    Table dataTable = dataSource.read();
    Table resultTable = this.algorithm.run(dataTable);
    IDataSource resultSource = new CsvDataSource(outputFilePath);
    resultSource.write(resultTable);
  }
}

package com.cy.ftamlirtp.adaptor.single;
import com.cy.ftamlirtp.core.common.algorithm.IMultiputAlgorithm;
import com.cy.ftamlirtp.core.common.datasource.CsvDataSource;
import com.cy.ftamlirtp.core.common.datasource.IDataSource;
import com.cy.ftamlirtp.core.common.table.Table;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class SingleMultiputAdaptor {
  private IMultiputAlgorithm algorithm;

  public void setAlgorithm(IMultiputAlgorithm algorithm) {
    this.algorithm = algorithm;
  }

  public void run(String[] inputFiles, String[] outputFiles) throws IOException {
    List<Table> tables = new ArrayList<>();
    for (String inputFilePath : inputFiles) {
      IDataSource dataSource = new CsvDataSource(inputFilePath);
      tables.add(dataSource.read());
    }
    List<Table> resultTables = this.algorithm.run(tables);
    for (int i = 0; i < outputFiles.length && i < resultTables.size(); i++) {
      IDataSource resultSource = new CsvDataSource(outputFiles[i]);
      resultSource.write(resultTables.get(i));
    }
  }
}
package com.cy.ftamlirtp.core.algorithm.auto.regression;
import com.cy.ftamlirtp.core.common.algorithm.IAlgorithm;
import com.cy.ftamlirtp.core.common.table.Header;
import com.cy.ftamlirtp.core.common.table.Row;
import com.cy.ftamlirtp.core.common.table.Table;
import javafx.util.Pair;
import com.cy.ftamlirtp.classifiers.functions.LinearRegression;
import com.cy.ftamlirtp.core.Attribute;
import com.cy.ftamlirtp.core.DenseInstance;
import com.cy.ftamlirtp.core.Instance;
import com.cy.ftamlirtp.core.Instances;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

/**
 * 自回归修复不完整/不精确的值
 */
public class AutoRegression implements IAlgorithm{
    private Integer regressionSize;

    AutoRegression(Integer regressionSize){
        this.regressionSize = regressionSize;
    }

    private void println(String str){
        System.out.println(str);
    }

    private void print(String str){
        System.out.print(str);
    }

    public Table run(Table table){
        List tableAttrList = Arrays.asList(table.getHeader().toArray());
        if (tableAttrList.size() !=1 && tableAttrList.size()!= 2)
            throw new IllegalArgumentException("table columns error, expected 2 columns");
        List<Row> tableRowsList = table.getRows();
        int inputDataSize = tableRowsList.size();
        double[] attrData = new double[inputDataSize];
        boolean[] needRepair = new boolean[inputDataSize];

        // 求需要修复的点
        for (int i = 0; i < inputDataSize; i++){
            if(tableRowsList.get(i).get(0).equals("")){
                attrData[i] = 0.0;
                needRepair[i] = true;
            }
            else{
                attrData[i] = Double.parseDouble((String) tableRowsList.get(i).get(0));
                if(table.getHeader().size() == 2 && Integer.parseInt((String) tableRowsLi
st.get(i).get(1)) != 0)
                    needRepair[i] = true;
                else
                    needRepair[i] = false;
            }
        }
        // 返回值
        Header returnHeader = new Header(new String[] { (String)tableAttrList.get(0), "cl
eaned_"+tableAttrList.get(0)});
        List<Row> returnRows = new ArrayList<>();
        // 自回归训练集，由 ArrayList<Double> 回归 Double
        ArrayList<Pair<ArrayList<Double>, Double>> dataSet = new ArrayList<>();
        for(int i = 0; i < inputDataSize; i++){
            boolean isInstance = true;
            for(int j = 0; j <= regressionSize; j++){
                if(i + j >= inputDataSize || needRepair[i + j]){
                    isInstance = false;
                    break;
                }
            }
            if(isInstance){
                ArrayList<Double> X = new ArrayList<>();
                for(int j = 0; j < regressionSize; j++)
                    X.add(attrData[i+j]);
                dataSet.add(new Pair<>(X, attrData[i+regressionSize]));
            }
        }
        if(dataSet.size() > 0){
            LinearRegression linearRegression = new LinearRegression();
            ArrayList<Attribute> attrs = new ArrayList<>();
            for (int i = 0; i <= regressionSize; i++)
                attrs.add(new Attribute(""+i));
            Instances instances = new Instances("dataSet", attrs, dataSet.size());
            instances.setClassIndex(regressionSize);
            for(Pair<ArrayList<Double>, Double> pair : dataSet){
                Instance instance = new DenseInstance(regressionSize + 1);
                instance.setDataset(instances);
                for(int i = 0; i < pair.getKey().size(); i++)
                    instance.setValue(attrs.get(i), pair.getKey().get(i));
                instance.setValue(attrs.get(regressionSize), pair.getValue());
                instances.add(instance);
            }
            try{
                linearRegression.buildClassifier(instances);
            }
            catch (Exception e){
                System.out.println("train error! "+e.toString());
                return new Table(returnHeader, returnRows);
            }
            // 进行修复
            for (int i = 0; i < inputDataSize; i++){
                if (needRepair[i]){
                    Instance instance = new DenseInstance(regressionSize);
                    for(int j = 0; j < regressionSize; j++){
                        double data = i - regressionSize + j >= 0? attrData[i-regressionS
ize+j] : attrData[0];
                        instance.setValue(attrs.get(j), data);
                    }
                    try{
                        double repairedValue = linearRegression.classifyInstance(instance);

                        attrData[i] = repairedValue;
                    }
                    catch (Exception e){
                        System.out.println("test error! "+e.toString());
                        attrData[i] = 0.0;
                    }
                }
                returnRows.add(new Row(returnHeader, new String[]{
                        (String) tableRowsList.get(i).get(0),
                        String.format("%.3f", attrData[i])}, i));
            }
        }
        return new Table(returnHeader, returnRows);
    }
}
package com.cy.ftamlirtp.service.impl;
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;
import com.cy.ftamlirtp.mapper.ReportMapper;
import com.cy.ftamlirtp.domain.Report;
import com.cy.ftamlirtp.service.IReportService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;
@Service
public class ReportServiceImpl extends ServiceImpl<ReportMapper, Report> implements IRepo
rtService{
    @Autowired
    private ReportMapper reportMapper;

    @Override
    public Report getReportByReportId(Integer reportId){
        return reportMapper.selectReportByReportId(reportId);
    }

    @Override
    public List<Report> listReport(Report report){
        return reportMapper.selectReportList(report);
    }

    @Transactional
    @Override
    public int saveReport(Report report){
        return reportMapper.insertReport(report);
    }

    private void checkExisted(ReportEntity entity) {
        ReportEntity existEntity = getOne(
                Wrappers.lambdaQuery(ReportEntity.class)
                .eq(ReportEntity::getReportId, entity.getReportId)
                false);
        if (existEntity != null && !existEntity.getId().equals(entity.getId())) {
            throw new BizException("报告记录已存在！");
        }
    }

    @Override
    @Transactional
    public int delReportByReportIds(Integer[] reportIds){
        return reportMapper.deleteReportByReportIds(reportIds);
    }

    private void reportData(List<ReportDto> list) {
        List<String> list = list.stream().map(ReportDto::getReportId)
                .collect(Collectors.toList());
        list.stream().forEach(report -> {
            Report reportEntry = entryReportMap.get(report.getReportId);
            if (reportEntry != null) {
                report.setAlertId(ReportTrans(reportEntry.getAlertId));
                report.setDescription(ReportTrans(reportEntry.getDescription));
                report.setReportDate(ReportTrans(reportEntry.getReportDate));
            }
        });
    }

    @Override
    @Transactional
    public int updateReport(Report report){
        return reportMapper.updateReport(report);
    }

    @Override
    @Transactional
    public int delReportByReportId(Integer reportId){
        return reportMapper.deleteReportByReportId(reportId);
    }
}
package com.cy.ftamlirtp.domain;
import java.util.Date;
import com.fasterxml.jackson.annotation.JsonFormat;
import lombok.Data;
import com.cy.ftamlirtp.common.annotation.Excel;
import com.baomidou.mybatisplus.annotation.TableName;
import com.cy.ftamlirtp.common.core.domain.BaseEntity;
```

```
/**
 * 监控规则实体类
 */
@TableName("ftamlirtp_monitoringrule")
@Data
public class MonitoringRule extends BaseEntity {
    private static final long serialVersionUID = -2017022223767433198LL;
    // 规则 ID
    private Integer ruleId;
    // 更新时间
    private Date updatedAt;
    // 是否启用
    private Integer enabled;
    // 规则描述
    private String description;
    // 规则名称
    private String name;
    // 创建时间
    private Date createdAt;
    // 监控标准
    private String criteria;
}
package com.cy.ftamlirtp.domain;
import java.math.BigDecimal;
import java.util.Date;
import com.fasterxml.jackson.annotation.JsonFormat;
import lombok.Data;
import com.cy.ftamlirtp.common.annotation.Excel;
import com.baomidou.mybatisplus.annotation.TableName;
import com.cy.ftamlirtp.common.core.domain.BaseEntity;
/**
 * 交易记录实体类
 */
@TableName("ftamlirtp_transaction")
@Data
public class Transaction extends BaseEntity {
    private static final long serialVersionUID = --2673028274567327581LL;
    // 交易 ID
    private Integer transactionId;
    // 账户 ID
    private Integer accountId;
    // 交易金额
    private BigDecimal amount;
    // 交易发起地
    private String origin;
    // 交易描述
    private String description;
    // 更新时间
    private Date updatedAt;
    // 交易目的地
    private String destination;
    // 交易类型
    private String type;
    // 交易状态
    private String status;
    // 创建时间
    private Date createdAt;
    // 交易货币
    private String currency;
    // 交易日期
    private String transactionDate;
    // 可疑标志
    private Integer suspiciousFlag;
    // 是否已报告
    private Integer reported;
}
package com.cy.ftamlirtp.controller;
import java.util.List;
import javax.servlet.http.HttpServletResponse;
import com.cy.ftamlirtp.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.cy.ftamlirtp.common.annotation.Log;
import com.cy.ftamlirtp.common.core.controller.BaseController;
import com.cy.ftamlirtp.common.utils.poi.ExcelUtil;
import com.cy.ftamlirtp.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.cy.ftamlirtp.common.enums.BusinessType;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import com.cy.ftamlirtp.domain.Report;
import com.cy.ftamlirtp.service.IReportService;
import org.springframework.web.bind.annotation.PathVariable;
@RestController
@RequestMapping("/report")
public class ReportController extends BaseAction {
    @Autowired
    private IReportService reportService;

    /**
     * 查询报告列表
     */
    @GetMapping("/list")
    public TableDataInfo reportList(ReportQueryVo queryVo){
        startPage();
        List<Report> reportList = reportService.getReportList(queryVo);
        list.forEach(i -> {
            i.setAlertId(ReportConvUtil(i.getAlertId));
            i.setDescription(ReportConvUtil(i.getDescription));
            i.setReportDate(ReportConvUtil(i.getReportDate));
        });
        PageInfo<Report> pageInfo = new PageInfo<>(list);
        return RespResult.pageResult(list, pageInfo.getTotal());
    }

    @PostMapping
    public RespResult addReport(@Valid @RequestBody ReportAddVo addVo){
       reportService.saveReport(addVo);
       return RespResult.success();
    }

    public RespResult listreport(Report report){
        List<Report> list = reportService.getReportList(report);
        return RespResult.success(list);
    }

    @DeleteMapping("/{reportIds}")
    public RespResult delreport(@PathVariable Integer[] reportIds){
        reportService.deleteReportByReportIds(reportIds);
        return RespResult.success();
    }

    @PutMapping
    public RespResult updatereport(@Valid @RequestBody Report report){
        reportService.updateReport(report);
        return RespResult.success();
    }

    @PostMapping("/export")
    public void exportReport(HttpServletResponse response, Report report){
        List<Report> list = reportService.getReportList(report);
        list.forEach(i -> {
            i.setAlertId(ReportExportFormat(i.getAlertId));
            i.setDescription(ReportExportFormat(i.getDescription));
            i.setReportDate(ReportExportFormat(i.getReportDate));
        });
        ExcelUtil<Report> excelUtil = new ExcelUtil<Report>(Report.class);
        util.exportExcel(response, list, "报告数据");
    }

    @GetMapping(value = "/{reportId}")
    public RespResult getreportId(@PathVariable("reportId") Integer reportId){
        return RespResult.success(reportService.getReportByReportId(reportId));
    }
}
package com.cy.ftamlirtp.domain;
```

```
import java.util.Date;
import com.fasterxml.jackson.annotation.JsonFormat;
import lombok.Data;
import com.cy.ftamlirtp.common.annotation.Excel;
import com.baomidou.mybatisplus.annotation.TableName;
import com.cy.ftamlirtp.common.core.domain.BaseEntity;
/**
 * 报告实体类
 */
@TableName("ftamlirtp_report")
@Data
public class Report extends BaseEntity {
    private static final long serialVersionUID = --3270991362474555913LL;
    // 报告 ID
    private Integer reportId;
    // 报告类型
    private String reportType;
    // 报告日期
    private String reportDate;
    // 更新时间
    private Date updatedAt;
    // 关联的预警 ID
    private Integer alertId;
    // 报告状态
    private String status;
    // 创建时间
    private Date createdAt;
    // 报告描述
    private String description;
    // 报告接收方
    private String recipient;
}
<?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.cy.ftamlirtp.mapper.AccountMapper">
    <resultMap type="Account" id="AccountResult">
        <result property="accountId" column="account_ID"    />
        <result property="balance" column="balance"    />
        <result property="updatedAt" column="updated_at"    />
        <result property="closureDate" column="closure_date"    />
        <result property="openingDate" column="opening_date"    />
        <result property="accountType" column="account_type"    />
        <result property="accountNumber" column="account_number"    />
        <result property="accountStatus" column="account_status"    />
        <result property="createdAt" column="created_at"    />
        <result property="customerId" column="customer_ID"    />
        <result property="freezeStatus" column="freeze_status"    />
        <result property="limit" column="limit"    />
        <result property="currency" column="currency"    />
    </resultMap>
    <sql id="selectAccountVo">
        select account_ID, balance, updated_at, closure_date, opening_date, account_type,
account_number, account_status, created_at, customer_ID, freeze_status, limit, currency
        from ftamlirtp_account
    </sql>
    <select id="selectAccountList" parameterType="Account" resultMap="AccountResult">
        <include refid="selectAccountVo"/>
        <where>
            <if test="accountId != null ">
            and account_ID = #{accountId}
            </if>
            <if test="balance != null ">
            and balance = #{balance}
            </if>
            <if test="updatedAt != null ">
            and updated_at = #{updatedAt}
            </if>
            <if test="closureDate != null and closureDate != ''">
            and closure_date = #{closureDate}
            </if>
            <if test="openingDate != null and openingDate != ''">
            and opening_date = #{openingDate}
            </if>
            <if test="accountType != null and accountType != ''">
            and account_type = #{accountType}
            </if>
            <if test="accountNumber != null and accountNumber != ''">
            and account_number = #{accountNumber}
            </if>
            <if test="accountStatus != null and accountStatus != ''">
            and account_status = #{accountStatus}
            </if>
            <if test="createdAt != null ">
            and created_at = #{createdAt}
            </if>
            <if test="customerId != null ">
            and customer_ID = #{customerId}
            </if>
            <if test="freezeStatus != null ">
            and freeze_status = #{freezeStatus}
            </if>
            <if test="limit != null ">
            and limit = #{limit}
            </if>
            <if test="currency != null and currency != ''">
            and currency = #{currency}
            </if>
        </where>
    </select>
    <delete id="deleteAccountByAccountId" parameterType="Integer">
        delete from ftamlirtp_account where account_ID = #{accountId}
    </delete>

    <insert id="insertAccount" parameterType="Account" useGeneratedKeys="true" keyPropert
y="accountId">
        insert into ftamlirtp_account
        <trim prefix="(" suffix=")" suffixOverrides=",">
            <if test="balance != null">balance,</if>
            <if test="updatedAt != null">updated_at,</if>
            <if test="closureDate != null and closureDate != ''">closure_date,</if>
            <if test="openingDate != null and openingDate != ''">opening_date,</if>
            <if test="accountType != null and accountType != ''">account_type,</if>
            <if test="accountNumber != null and accountNumber != ''">account_number,</if>
            <if test="accountStatus != null and accountStatus != ''">account_status,</if>
            <if test="createdAt != null">created_at,</if>
            <if test="customerId != null">customer_ID,</if>
            <if test="freezeStatus != null">freeze_status,</if>
            <if test="limit != null">limit,</if>
            <if test="currency != null and currency != ''">currency,</if>
         </trim>
        <trim prefix="values (" suffix=")" suffixOverrides=",">
            <if test="balance != null">#{balance},</if>
            <if test="updatedAt != null">#{updatedAt},</if>
            <if test="closureDate != null and closureDate != ''">#{closureDate},</if>
            <if test="openingDate != null and openingDate != ''">#{openingDate},</if>
            <if test="accountType != null and accountType != ''">#{accountType},</if>
            <if test="accountNumber != null and accountNumber != ''">#{accountNumber},</i
f>
            <if test="accountStatus != null and accountStatus != ''">#{accountStatus},</i
f>
            <if test="createdAt != null">#{createdAt},</if>
            <if test="customerId != null">#{customerId},</if>
            <if test="freezeStatus != null">#{freezeStatus},</if>
            <if test="limit != null">#{limit},</if>
            <if test="currency != null and currency != ''">#{currency},</if>
         </trim>
    </insert>
    <delete id="deleteAccountByAccountIds" parameterType="String">
        delete from ftamlirtp_account where account_ID in
        <foreach item="accountId" collection="array" open="(" separator="," close=")">
            #{accountId}
        </foreach>
    </delete>
</mapper>
package com.cy.ftamlirtp.service.impl;
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;
import com.cy.ftamlirtp.mapper.AlertRecordMapper;
import com.cy.ftamlirtp.domain.AlertRecord;
import com.cy.ftamlirtp.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.setAlertDate(AlertRecordTrans(alertRecordEntry.getAlertDate));
                alertRecord.setTransactionId(AlertRecordTrans(alertRecordEntry.getTransac
tionId));
            }
        });
    }

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

    @Override
    @Transactional
    public int delAlertRecordByAlertId(Integer alertId){
        return alertRecordMapper.deleteAlertRecordByAlertId(alertId);
    }
}
package com.cy.ftamlirtp.service.sms.controller;
import com.aliyuncs.exceptions.ClientException;
import com.cy.ftamlirtp.common.base.result.R;
import com.cy.ftamlirtp.common.base.result.ResultCode;
import com.cy.ftamlirtp.common.base.util.FormUtils;
import com.cy.ftamlirtp.common.base.util.RandomUtils;
import com.cy.ftamlirtp.service.base.exception.GuridException;
import com.cy.ftamlirtp.service.sms.service.SmsService;
import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.Parameter;
import io.swagger.v3.oas.annotations.tags.Tag;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.redis.core.RedisTemplate;
import org.springframework.util.StringUtils;
import org.springframework.web.bind.annotation.*;
import java.util.concurrent.TimeUnit;
@RestController
@RequestMapping("/api/sms")
@Tag(name = "ApiSmsController", description = "短信服务")
@Slf4j
public class ApiSmsController {
    @Autowired
    private SmsService smsService;

    @Autowired
    private RedisTemplate redisTemplate;

    @Operation(description = "获取验证码")
    @GetMapping("send/{mobile}")
    public R getCode(@Parameter(description = "手机号", required = true) @PathVariable St
ring mobile) throws ClientException {
        if (!(StringUtils.hasText(mobile)
                && mobile.length() == 11
                && FormUtils.isMobile(mobile))) {
            log.error("手机号不正确");
            throw new GuridException(ResultCode.LOGIN_PHONE_ERROR);
        }
        String checkCode = RandomUtils.getSixBitRandom();
        System.out.println(checkCode);
        //smsService.send(mobile, checkCode);
        redisTemplate.opsForValue().set(mobile, checkCode, 5, TimeUnit.MINUTES);
        return R.ok().message("短信发送成功");
    }
}
package com.cy.ftamlirtp.service.sms.service;
import com.aliyuncs.exceptions.ClientException;
public interface SmsService {
    void send(String mobile, String checkCode) throws ClientException;
}
package com.cy.ftamlirtp.service.sms.service.impl;
import com.aliyun.dysmsapi20170525.models.SendSmsResponse;
import com.aliyuncs.CommonRequest;
import com.aliyuncs.CommonResponse;
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.IAcsClient;
import com.aliyuncs.exceptions.ClientException;
import com.aliyuncs.http.MethodType;
import com.aliyuncs.profile.DefaultProfile;
import com.google.gson.Gson;
import com.cy.ftamlirtp.common.base.result.ResultCode;
import com.cy.ftamlirtp.service.base.exception.GuridException;
import com.cy.ftamlirtp.service.sms.service.SmsService;
import com.cy.ftamlirtp.service.sms.util.AliyunSmsUtils;
import com.cy.ftamlirtp.service.sms.util.SmsProperties;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.HashMap;
import java.util.Map;
@Service
public class SmsServiceImpl implements SmsService {

    private static final Logger log = LoggerFactory.getLogger(SmsServiceImpl.class);
    @Autowired
    private SmsProperties properties;

    @Override
    public void send(String mobile, String checkCode) throws ClientException {
        DefaultProfile profile = DefaultProfile.getProfile(properties.getRegionId(), prop
erties.getKeyId(), properties.getKeySecrete());
        IAcsClient client = new DefaultAcsClient(profile);
        CommonRequest request = new CommonRequest();
        request.setSysMethod(MethodType.POST);
        request.setSysDomain(properties.getEndpoint());
        request.setSysVersion("2017-05-25");
        request.setSysAction("SendSms");
        request.putQueryParameter("RegionId", properties.getRegionId());
        request.putQueryParameter("PhoneNumbers", mobile);
        request.putQueryParameter("SignName", properties.getSignName());
        request.putQueryParameter("TemplateCode", properties.getTemplateCode());
        Map<String, String> params = new HashMap<String, String>();
        params.put("Code", checkCode);
        Gson gson = new Gson();
        String json = gson.toJson(params);
        request.putQueryParameter("TemplateParam", json);
        CommonResponse response = client.getCommonResponse(request);
        System.out.println(response.getData());
        HashMap<String, String> map = gson.fromJson(response.getData(), HashMap.class);
        String code = map.get("Code");
        String message = map.get("Message");
        if ("isv.BUSINESS_LIMIT_CONTROL".equals(code)) {
            log.error("发送短信过于频繁" + "code-" + code + "message-" + message);
            throw new GuridException(ResultCode.SMS_SEND_ERROR_BUSINESS_LIMIT_CONTROL);
        }
        if (!"OK".equals(code)) {
            log.error("短信发送失败" + "code-" + code + "message-" + message);
            throw new GuridException(ResultCode.SMS_SEND_ERROR);
        }

    }

    public void send_new(String mobile, String checkCode) throws Exception {
        Map<String, String> params = new HashMap<String, String>();
        params.put("Code", checkCode);
        Gson gson = new Gson();
        String json = gson.toJson(params);
        SendSmsResponse response = AliyunSmsUtils.sendSms(
                properties.getKeyId(),
                properties.getKeySecrete(),
                properties.getEndpoint(),
                mobile,
                properties.getSignName(),
                properties.getTemplateCode(),
                json);

        //解析响应结果
        HashMap<String, String> map = gson.fromJson(String.valueOf(response.body), HashMa
p.class);
        String code = map.get("Code");
        String message = map.get("Message");
        if ("isv.BUSINESS_LIMIT_CONTROL".equals(code)) {
            log.error("发送短信过于频繁" + "code-" + code + "message-" + message);
            throw new GuridException(ResultCode.SMS_SEND_ERROR_BUSINESS_LIMIT_CONTROL);
        }

        if (!"OK".equals(code)) {
            log.error("短信发送失败" + "code-" + code + "message-" + message);
            throw new GuridException(ResultCode.SMS_SEND_ERROR);
        }
    }
}
package com.cy.ftamlirtp.util;
import java.io.File;
import java.net.URI;
import java.util.HashMap;
import java.util.Map;
import java.util.Properties;
import org.springframework.beans.BeansException;
import org.springframework.beans.factory.config.ConfigurableListableBeanFactory;
import org.springframework.beans.factory.config.PropertyPlaceholderConfigurer;
import org.springframework.context.ApplicationContext;
import org.springframework.context.ApplicationContextAware;
/**
 * 资源工具类
 */
public class ResourceUtil extends PropertyPlaceholderConfigurer implements ApplicationCon
textAware {

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

    public static String toString(InputStream input, String encoding) throws IOException
{
        BufferedReader br = null;
        try {
            StringBuilder sb = new StringBuilder();
            br = new BufferedReader(new InputStreamReader(input, encoding));
            String line;
            while ((line = br.readLine()) != null) {
                sb.append(line).append("\n");
            }
            return sb.toString();
        } finally {
            if (br != null) {
                try {
                    br.close();
                } catch (IOException ignored) {
                }
            }
        }
    }
}
package com.cy.ftamlirtp.controller;
import com.cy.ftamlirtp.service.AlarmService;
import com.cy.ftamlirtp.service.StationService;
import com.cy.ftamlirtp.util.ListObject;
import com.cy.ftamlirtp.util.ResponseUtil;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestHeader;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import java.text.ParseException;
@RestController
```

```
public class StationController {
    @Autowired
    private StationService stationService;
    @Autowired
    private AlarmService alarmService;

    /**
     * 获取站点树
     * @param userId 用户 id
     * @return ListObject
     */
    @RequestMapping("/getStationTree")
    public ListObject getStationTree(@RequestHeader(value = "currentId")String userId){
        return ResponseUtil.queryListObject(stationService.queryStationTree(userId),"获取
站点树成功");
    }

    /**
     * 获取站点实时排名
     * @param userId 用户 id
     * @param pageNum 页码
     * @param pageSize 页大小
     * @param sortType 排序
     * @return ListObject
     */
    @RequestMapping("/getStationRanking")
    public ListObject getStationRanking(@RequestHeader(value = "currentId")String userId,
                                    @RequestParam(value = "pageNum",required = false)
String pageNum,
                                    @RequestParam(value = "pageSize",required = false)S
tring pageSize,
                                    @RequestParam(value = "sortType",required = false)S
tring sortType){
        return ResponseUtil.queryListObject(stationService.getStationRanking(userId,pageN
um,pageSize,sortType),"获取站点树成功");
    }

    /**
     * 获取报警记录
     * @param siteId 站点 id
     * @param stationId 监测点 id
     * @param alarmType 报警类型
     * @return ListObject
     */
    @RequestMapping("/getAlarmRecord")
    public ListObject getAlarmRecord(@RequestParam(value = "siteId")String siteId,
                                 @RequestParam(value = "stationId")String stationId,
                                 @RequestParam(value = "alarmType")String alarmType,
                                 @RequestParam(value = "pageNum")String pageNum,
                                 @RequestParam(value = "pageSize")String pageSize) th
```

```
rows ParseException{
        return ResponseUtil.queryListObject(alarmService.getAlarmRecord(siteId,stationId,
alarmType,pageNum,pageSize),"成功");
    }

    /**
     * 删除报警记录
     * @param alarmId 报警记录 id
     * @return ListObject
     */
    @RequestMapping("/deleteAlarmRecord")
    public ListObject deleteAlarmRecord(@RequestParam("alarmId")String alarmId){
        alarmService.deleteAlarmRecord(alarmId);
        return ResponseUtil.success();
    }

    /**
     * 获取报警规则列表
     * @param siteId 站点 id
     * @param stationId 监测点 id
     * @param alarmType 报警类型
     * @param pageNum 页码
     * @param pageSize 页大小
     * @return ListObject
     */
    @RequestMapping("/getStandardList")
    public ListObject getStandardList(@RequestParam(value = "siteId")String siteId,
                                  @RequestParam(value = "stationId")String stationId,
                                  @RequestParam(value = "alarmType")String alarmType,
                                  @RequestParam(value = "pageNum")String pageNum,
                                  @RequestParam(value = "pageSize")String pageSize){
        return ResponseUtil.queryListObject(alarmService.getStandardList(siteId,stationId,a
larmType,pageNum,pageSize),"成功");
    }


    /**
     * 获取站点对应因子
     * @return ListObject
     */
    @RequestMapping("/getStationProject")
    public ListObject getStationProject(@RequestHeader(value = "currentId")String userId)
{
        return ResponseUtil.queryListObject(alarmService.getStationProject(userId),"成功");

    }

    /**
     * 获取报警人员
     * @return ListObject
     */
    @RequestMapping("/getAlarmPerson")
    public ListObject getAlarmPerson(){
        return ResponseUtil.queryListObject(alarmService.getAlarmPerson(),"成功");
    }

    @RequestMapping("/setAlarmStandard")
    public ListObject setAlarmStandard(@RequestParam(value = "alarmId",required = false)S
tring alarmId,
                                       @RequestParam(value = "stationId")String stationId,
                                       @RequestParam(value = "alarmType")String alarmType,
                                       @RequestParam(value = "alarmLevel",required = fals
e)String alarmLevel,
                                       @RequestParam(value = "alarmStep",required = false)S
tring alarmStep,
                                       @RequestParam(value = "alarmPerson")String alarmPe
rson,
                                       @RequestParam(value = "workDay")String workDay,
                                       @RequestParam(value = "time")String time,
                                       @RequestParam(value = "interval")String interval){
        int num = alarmService.setAlarmStandard(alarmId,stationId,alarmType,alarmLevel,al
armStep,alarmPerson,workDay,time,interval);
        if(num==-1){
            return ResponseUtil.error("同一个站点同一种规则只能添加一个!");
        }
        return ResponseUtil.success();
    }

    /**
     * 删除报警规则
     * @param alarmId 规则 id
     * @return ListObject
     */
    @RequestMapping("/deleteAlarmStandard")
    public ListObject deleteAlarmStandard(@RequestParam("alarmId")String alarmId){
        alarmService.deleteAlarmStandard(alarmId);
        return ResponseUtil.success();
    }
}
package com.cy.ftamlirtp.controller;
import com.cy.ftamlirtp.common.constants.Constants;
import com.cy.ftamlirtp.common.entity.HeraHostRelation;
import com.cy.ftamlirtp.common.entity.HeraJobMonitor;
import com.cy.ftamlirtp.common.entity.model.JsonResponse;
import com.cy.ftamlirtp.common.entity.model.TableResponse;
import com.cy.ftamlirtp.common.entity.vo.HeraActionVo;
import com.cy.ftamlirtp.common.entity.vo.HeraJobMonitorVo;
import com.cy.ftamlirtp.common.entity.vo.HeraSsoVo;
import com.cy.ftamlirtp.common.exception.NoPermissionException;
import com.cy.ftamlirtp.common.service.HeraHostRelationService;
import com.cy.ftamlirtp.common.service.HeraJobActionService;
import com.cy.ftamlirtp.common.service.HeraJobMonitorService;
import com.cy.ftamlirtp.common.service.HeraSsoService;
import com.cy.ftamlirtp.config.AdminCheck;
import com.cy.ftamlirtp.core.netty.worker.WorkClient;
import com.cy.ftamlirtp.monitor.service.JobManageService;
import com.cy.ftamlirtp.protocol.JobExecuteKind;
import com.google.protobuf.InvalidProtocolBufferException;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import io.swagger.annotations.ApiParam;
import org.apache.commons.lang.StringUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;
import java.util.*;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.TimeoutException;
@Controller
@Api("系统操作接口")
public class SystemManageController extends BaseHeraController {

    @Autowired
    private JobManageService jobManageService;

    @Autowired
    private HeraJobActionService heraJobActionService;

    @Autowired
    private HeraHostRelationService heraHostRelationService;

    @Autowired
    private WorkClient workClient;

    @Autowired
    private HeraJobMonitorService heraJobMonitorService;

    @Autowired
    private HeraSsoService heraSsoService;


    @GetMapping("/userManage")
    @AdminCheck
    public String userManage() throws NoPermissionException {
        return "systemManage/userManage.index";
    }

    @GetMapping("/basicManage")
    @AdminCheck
    public String basicManage() throws NoPermissionException {
        return "systemManage/basicManage.index";
    }

    @RequestMapping("/workManage")
    @GetMapping
    public String workManage() throws NoPermissionException {
        return "systemManage/workManage.index";
    }



    @GetMapping("/hostGroupManage")
    @AdminCheck
    public String hostGroupManage() throws NoPermissionException {
        return "systemManage/hostGroupManage.index";
    }

    @GetMapping("/jobMonitor")
    @AdminCheck
    public String jobMonitor() throws NoPermissionException {
        return "systemManage/jobMonitor.index";
    }

    @GetMapping("/jobDetail")
    public String jobManage() {
        return "jobManage/jobDetail.index";
    }



    @GetMapping("/rerun")
    public String jobRerun() {
        return "jobManage/rerun.index";
    }

    @GetMapping("/jobSearch")
    public String jobSearch() {
        return "jobManage/jobSearch.index";
    }



    @GetMapping("/jobDag")
    public String jobDag() {
        return "jobManage/jobDag.index";
    }

    @GetMapping("/machineInfo")
    public String machineInfo() {
        return "machineInfo";
    }

    @RequestMapping(value = "/workManage/list", method = RequestMethod.GET)
    @ResponseBody
    @AdminCheck
    @ApiOperation("机器组关系列表查询")
    public TableResponse workManageList() {
        List<HeraHostRelation> hostRelations = heraHostRelationService.getAll();
        if (hostRelations == null) {
            return new TableResponse(-1, "查询失败");
        }
        return new TableResponse(hostRelations.size(), 0, hostRelations);
    }

    @RequestMapping(value = "/workManage/add", method = RequestMethod.POST)
    @ResponseBody
    @AdminCheck
    @ApiOperation("机器组关系添加")
    public JsonResponse workManageAdd(@ApiParam(value = "机器组管理对象",required = true)
HeraHostRelation heraHostRelation) {
        int insert = heraHostRelationService.insert(heraHostRelation);
        if (insert > 0) {
            return new JsonResponse(true, "插入成功");
        }
        return new JsonResponse(false, "插入失败");

    }

    @RequestMapping(value = "/workManage/del", method = RequestMethod.POST)
    @ResponseBody
    @AdminCheck
    @ApiOperation("机器组关系删除")
    public JsonResponse workManageDel(@ApiParam(value = "机器组关系 id",required = true)In
teger id) {
        int delete = heraHostRelationService.delete(id);
        if (delete > 0) {
            return new JsonResponse(true, "删除成功");
        }
        return new JsonResponse(false, "删除失败");
    }

    @RequestMapping(value = "/workManage/update", method = RequestMethod.POST)
    @ResponseBody
    @AdminCheck
    @ApiOperation("机器组关系更新")
    public JsonResponse workManageUpdate(@ApiParam(value = "机器组管理对象",required = tr
ue)HeraHostRelation heraHostRelation) {
        int update = heraHostRelationService.update(heraHostRelation);
        if (update > 0) {
            return new JsonResponse(true, "更新成功");
        }
        return new JsonResponse(false, "更新失败");
    }

    @GetMapping(value = "/jobMonitor/list")
    @ResponseBody
    @ApiOperation("任务监控列表查询")
    public TableResponse jobMonitorList() {
        List<HeraJobMonitorVo> monitors = heraJobMonitorService.findAllVo();
        if (monitors == null || monitors.size() == 0) {
            return new TableResponse(-1, "无监控任务");
        }
        Map<String, HeraSsoVo> cacheSso = new HashMap<>();
        monitors.forEach(monitor -> {
            if (!StringUtils.isBlank(monitor.getUserIds())) {
                List<HeraSsoVo> ssoVos = new ArrayList<>();
                Arrays.stream(monitor.getUserIds().split(Constants.COMMA)).filter(StringU
tils::isNotBlank).distinct().forEach(id -> {
                    HeraSsoVo ssoVo = cacheSso.get(id);
                    if (ssoVo == null) {
                        ssoVo = heraSsoService.findSsoVoById(Integer.parseInt(id));
                        cacheSso.put(id, ssoVo);
                    }
                    ssoVos.add(ssoVo);
                });
                monitor.setMonitors(ssoVos);
            } else {
                monitor.setMonitors(new ArrayList<>(0));
            }
            Optional.ofNullable(monitor.getUserIds()).ifPresent(userIds -> {
                if (userIds.endsWith(Constants.COMMA)) {
                    monitor.setUserIds(userIds.substring(0, userIds.length() - 1));
                }
            });
        });
        cacheSso.clear();
        return new TableResponse(monitors.size(), 0, monitors);
    }

    @PostMapping(value = "/jobMonitor/add")
    @ResponseBody
    @AdminCheck
    @ApiOperation("任务监控人添加接口")
    public JsonResponse jobMonitorAdd(@ApiParam(value = "任务 id",required = true)Integer
jobId,
                                      @ApiParam(value = "监控人 hera_sso 的 id 集合,多个,分
割",required = true)String monitors) {
        HeraJobMonitor monitor = heraJobMonitorService.findByJobId(jobId);
        if (monitor != null) {
            return new JsonResponse(false, "该监控任务已经存在,请直接编辑该任务");
        }
        boolean res = heraJobMonitorService.addMonitor(monitors, jobId);
        return new JsonResponse(res, res ? "添加监控成功" : "添加监控失败");
    }

    @PostMapping(value = "/jobMonitor/update")
    @ResponseBody
    @AdminCheck
    @ApiOperation("任务监控人更新接口")
    public JsonResponse jobMonitorUpdate(@ApiParam(value = "任务 id",required = true)Integ
er jobId,
                                         @ApiParam(value = "监控人 hera_sso 的 id 集合,多个,
分割",required = true)String monitors) {
        boolean res = heraJobMonitorService.updateMonitor(monitors, jobId);
        return new JsonResponse(res, res ? "添加监控成功" : "添加监控失败");
    }

    /**
     * 首页任务运行 top10
     *
     * @return
     */
    @RequestMapping(value = "/homePage/findJobRunTimeTop10", method = RequestMethod.GET)
    @ResponseBody
    @ApiOperation("首页任务运行 top10")
    public JsonResponse findJobRunTimeTop10() {
        return jobManageService.findJobRunTimeTop10();
    }


    /**
     * 今日所有任务状态，初始化首页饼图
     *
     * @return
     */
    @RequestMapping(value = "/homePage/findAllJobStatus", method = RequestMethod.GET)
    @ResponseBody
    @ApiOperation("今日所有任务状态，初始化首页饼图")
    public JsonResponse findAllJobStatus() {
        return jobManageService.findAllJobStatus();
    }


    /**
     * 今日所有任务状态明细，线形图初始化
     *
     * @return
     */
    @RequestMapping(value = "/homePage/findAllJobStatusDetail", method = RequestMethod.GE
T)
    @ResponseBody
    @ApiOperation("今日任务详情明细，初始化曲线图")
    public JsonResponse findAllJobStatusDetail() {
        return jobManageService.findAllJobStatusDetail();
    }


    @RequestMapping(value = "/homePage/getJobQueueInfo", method = RequestMethod.GET)
    @ResponseBody
    @ApiOperation("获取任务在 work/master 上的等待/执行队列及监控信息")
    public JsonResponse getJobQueueInfo() throws InterruptedException, ExecutionException,
InvalidProtocolBufferException, TimeoutException {
        return new JsonResponse(true, workClient.getJobQueueInfoFromWeb());

    }


    @RequestMapping(value = "/homePage/getNotRunJob", method = RequestMethod.GET)
    @ResponseBody
    @ApiOperation("查询未执行的任务")
    public JsonResponse getNotRunJob() {
        List<HeraActionVo> scheduleJob = heraJobActionService.getNotRunScheduleJob();
        return new JsonResponse(true, "查询成功", scheduleJob);
    }

    @RequestMapping(value = "/homePage/getFailJob", method = RequestMethod.GET)
    @ResponseBody
    @ApiOperation("查询最后一次调度失败的任务")
    public JsonResponse getScheduleFailJob() {
        List<HeraActionVo> failedJob = heraJobActionService.getFailedJob();
        return new JsonResponse(true, "查询成功", failedJob);
    }

    @RequestMapping(value = "/homePage/getAllWorkInfo", method = RequestMethod.GET)
    @ResponseBody
    @ApiOperation("查询所有的机器监控信息")
    public JsonResponse getAllWorkInfo() throws InterruptedException, ExecutionException,
InvalidProtocolBufferException, TimeoutException {
        return new JsonResponse(true, workClient.getAllWorkInfo());
    }


    @RequestMapping(value = "/isAdmin", method = RequestMethod.GET)
    @ResponseBody
    public JsonResponse checkAdmin() {
        return new JsonResponse(true, isAdmin());
    }


    @RequestMapping(value = "/admin/generateAllVersion", method = RequestMethod.PUT)
    @ResponseBody
    @AdminCheck
    @ApiOperation("全量版本生成接口")
    public JsonResponse generateAllVersion() throws ExecutionException, InterruptedExcept
ion, TimeoutException {
        return new JsonResponse(true, workClient.generateActionFromWeb(JobExecuteKind.Exe
cuteKind.ManualKind, Constants.ALL_JOB_ID));
    }

    @RequestMapping(value = "admin/updateWork", method = RequestMethod.PUT)
    @ResponseBody
    @AdminCheck
    @ApiOperation("更新 work 信息触发接口")
    public JsonResponse updateWork() throws ExecutionException, InterruptedException, Tim
eoutException {
        return new JsonResponse(true, workClient.updateWorkFromWeb());
    }
}
package com.cy.ftamlirtp.service.impl;
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;
import com.cy.ftamlirtp.mapper.CustomerMapper;
import com.cy.ftamlirtp.domain.Customer;
import com.cy.ftamlirtp.service.ICustomerService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;
@Service
public class CustomerServiceImpl extends ServiceImpl<CustomerMapper, Customer> implements
ICustomerService{
    @Autowired
    private CustomerMapper customerMapper;

    @Override
    public Customer getCustomerByCustomerId(Integer customerId){
        return customerMapper.selectCustomerByCustomerId(customerId);
    }

    @Override
    public List<Customer> listCustomer(Customer customer){
        return customerMapper.selectCustomerList(customer);
    }

    @Transactional
    @Override
    public int saveCustomer(Customer customer){
        return customerMapper.insertCustomer(customer);
    }

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

    @Override
    @Transactional
    public int delCustomerByCustomerIds(Integer[] customerIds){
        return customerMapper.deleteCustomerByCustomerIds(customerIds);
    }

    private void customerData(List<CustomerDto> list) {
        List<String> list = list.stream().map(CustomerDto::getCustomerId)
                .collect(Collectors.toList());
        list.stream().forEach(customer -> {
            Customer customerEntry = entryCustomerMap.get(customer.getCustomerId);
            if (customerEntry != null) {
                customer.setLastReviewDate(CustomerTrans(customerEntry.getLastReviewDate));

                customer.setOnboardDate(CustomerTrans(customerEntry.getOnboardDate));
                customer.setStatus(CustomerTrans(customerEntry.getStatus));
                customer.setDateOfBirth(CustomerTrans(customerEntry.getDateOfBirth));
                customer.setRiskLevel(CustomerTrans(customerEntry.getRiskLevel));
                customer.setUpdatedAt(CustomerTrans(customerEntry.getUpdatedAt));
                customer.setNextReviewDate(CustomerTrans(customerEntry.getNextReviewDate));

                customer.setCreatedAt(CustomerTrans(customerEntry.getCreatedAt));
                customer.setPhoneNumber(CustomerTrans(customerEntry.getPhoneNumber));
                customer.setName(CustomerTrans(customerEntry.getName));
                customer.setNationality(CustomerTrans(customerEntry.getNationality));
            }
        });
    }

    @Override
    @Transactional
    public int updateCustomer(Customer customer){
        return customerMapper.updateCustomer(customer);
    }

    @Override
    @Transactional
    public int delCustomerByCustomerId(Integer customerId){
        return customerMapper.deleteCustomerByCustomerId(customerId);
    }
}
package com.cy.ftamlirtp.utils;
/**
 * Base64 工具类
 */
public class Base64Util {
    private static final char last2byte = (char) Integer.parseInt("00000011", 2);
    private static final char last4byte = (char) Integer.parseInt("00001111", 2);
    private static final char last6byte = (char) Integer.parseInt("00111111", 2);
    private static final char lead6byte = (char) Integer.parseInt("11111100", 2);
    private static final char lead4byte = (char) Integer.parseInt("11110000", 2);
    private static final char lead2byte = (char) Integer.parseInt("11000000", 2);
    private static final char[] encodeTable = new char[]{'A', 'B', 'C', 'D', 'E', 'F', 'G
',
    'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X',
    'Y', 'Z', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o',
    'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', '0', '1', '2', '3', '4', '5',
    '6', '7', '8', '9', '+', '/'};

    public Base64Util() {
    }

    public static String encode(byte[] from) {
        StringBuilder to = new StringBuilder((int) ((double) from.length * 1.34D) + 3);
        int num = 0;
        char currentByte = 0;
        int i;
        for (i = 0; i < from.length; ++i) {
            for (num %= 8; num < 8; num += 6) {
                switch (num) {
                    case 0:
                        currentByte = (char) (from[i] & lead6byte);
                        currentByte = (char) (currentByte >>> 2);
                    case 1:
                    case 3:
                    case 5:
                    default:
                        break;
                    case 2:
                        currentByte = (char) (from[i] & last6byte);
                        break;
                    case 4:
                        currentByte = (char) (from[i] & last4byte);
                        currentByte = (char) (currentByte << 2);
                        if (i + 1 < from.length) {
                            currentByte = (char) (currentByte | (from[i + 1] & lead2byte)
>>> 6);
                        }
                        break;
                    case 6:
                        currentByte = (char) (from[i] & last2byte);
                        currentByte = (char) (currentByte << 4);
                        if (i + 1 < from.length) {
                            currentByte = (char) (currentByte | (from[i + 1] & lead4byte)
>>> 4);
                        }
                }
                to.append(encodeTable[currentByte]);
            }
        }

        if (to.length() % 4 != 0) {
            for (i = 4 - to.length() % 4; i > 0; --i) {
                to.append("=");
            }
        }
        return to.toString();
    }
}
package com.cy.ftamlirtp.utils;
import org.apache.commons.lang.StringUtils;
import org.apache.commons.lang.math.NumberUtils;
import java.math.BigDecimal;
import java.time.Duration;
import java.time.Instant;
import java.time.LocalDateTime;
import java.time.temporal.ChronoUnit;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.StringTokenizer;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
/**
 * 通用工具类
 */
public class CommonUtil {
    public static String hiddenMobile(String mobile) {
        if (StringUtils.isBlank(mobile)) {
            return "";
        }
        return mobile.replaceAll("(\\d{3})\\d{4}(\\d{4})","$1****$2");
    }
    /**
     * @param s
     * @param isFen
     * @return
     */
    public static BigDecimal strToBigDecimal(String s, Boolean isFen) {
        if (StringUtils.isBlank(s)) {
            return null;
        }
        if (!NumberUtils.isNumber(s)) {
            return null;
        }
        BigDecimal bd = new BigDecimal(s);
        if (isFen != null && isFen.booleanValue()){
            bd = bd.divide(new BigDecimal(100), 2);
        }
        return bd;
    }
    public static final Pattern shiFenMiaoPattern = Pattern.compile("(\\d{1,2}[：:]){0,2}
\\d{1,2}");
    public static Long shiFenMiaoToSeconds (String shifenmiao) {
        if (StringUtils.isBlank(shifenmiao)) {
            return 0L;
        }
        Long totalSeconds = 0L;
        shifenmiao = shifenmiao.replaceAll(" ", "");
        boolean matched = shiFenMiaoPattern.matcher(shifenmiao).matches();
        if (matched) {
            List<String> sfmList = new ArrayList<>();
            StringTokenizer st = new StringTokenizer(shifenmiao,":： ");
            while (st.hasMoreTokens()) {
                sfmList.add(st.nextToken());
            }
            Collections.reverse(sfmList);
            String[] sfmArr = sfmList.toArray(new String[0]);
            for (int i = 0; i < sfmArr.length; i++) {
                if (i == 0) {
                    totalSeconds += Long.valueOf(sfmArr[i]);
                } else if (i == 1) {
                    totalSeconds += Long.valueOf(sfmArr[i]) * 60;
                } else if (i == 2) {
                    totalSeconds += Long.valueOf(sfmArr[i]) * 3600;
                }
            }
        }
        return totalSeconds;
    }

    /**
     * 将下划线映射到骆驼命名使用的正则表达式，预编译正则用于提高效率
     */
    private static Pattern patternForUTC = Pattern.compile("_([a-z]){1}");

    /**
     * 将下划线映射到骆驼命名
     *
     * @param str
     * @return
     */
    public static String mapUnderscoreToCamelCase(String str) {
        str = str.toLowerCase();
        final Matcher matcher = patternForUTC.matcher(str);
        while (matcher.find()) {
            str = str.replaceAll(matcher.group(), matcher.group(1).toUpperCase());
        }
        return str;
    }

    /**
     * 将骆驼命名映射到下划线，必须是标准的驼峰命名，否则会出现奇怪的结果
     *
     * @param str
     * @return
     */
    public static String mapCamelCaseToUnderscore(String str) {
        return str.replaceAll("([A-Z]){1}","_$1").toUpperCase();
    }
}

package com.cy.ftamlirtp.common;
import java.lang.annotation.Documented;
import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;
import com.cy.ftamlirtp.common.BusinessType;
import com.cy.ftamlirtp.common.OperatorType;
@Target({ ElementType.PARAMETER, ElementType.METHOD })
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface Log{
    public String title() default "";
    public BusinessType businessType() default BusinessType.OTHER;
    public OperatorType operatorType() default OperatorType.MANAGE;
    public boolean isSaveRequestData() default true;
    public boolean isSaveResponseData() default true;
}
package com.cy.ftamlirtp.common.readingstorm2;
import org.apache.kafka.clients.consumer.ConsumerConfig;
import org.apache.storm.Config;
import org.apache.storm.LocalCluster;
import org.apache.storm.StormSubmitter;
import org.apache.storm.generated.AlreadyAliveException;
import org.apache.storm.generated.AuthorizationException;
import org.apache.storm.generated.InvalidTopologyException;
import org.apache.storm.kafka.spout.KafkaSpout;
import org.apache.storm.kafka.spout.KafkaSpoutConfig;
import org.apache.storm.kafka.spout.KafkaSpoutRetryExponentialBackoff;
import org.apache.storm.kafka.spout.KafkaSpoutRetryExponentialBackoff.TimeInterval;
import org.apache.storm.kafka.spout.KafkaSpoutRetryService;
import org.apache.storm.topology.TopologyBuilder;
import java.util.logging.Logger;
public class ReadingFromApp {

    private static final String BOOTSTRAP_SERVERS = "192.168.91.3:9092";
    private static final String TOPIC_NAME = "readingtopic0";
    private static final Logger logger = Logger.getLogger(ReadingFromApp.class.getName());
    static {
        Logger parentLogger = logger.getParent();
        parentLogger.setLevel(java.util.logging.Level.OFF);
    }
    public static void transConvert() throws Exception {
        final TopologyBuilder builder = new TopologyBuilder();
        builder.setSpout("kafka_spout", new Spout<>(getKafkaSpoutConfig(BOOTSTRAP_SERVERS,
TOPIC_NAME)), 1);
        builder.setBolt("split_bolt", new SplitBolt(), 2).shuffleGrouping("reading_spout").
setNumTasks(2);
        builder.setBolt("stat_store_bolt", new StatAndStoreBolt(), 1).shuffleGrouping("sp
lit_bolt").setNumTasks(1);
        StormSubmitter.submitTopology("ClusterReadingFromApp", new Config(), builder.crea
teTopology());
        if (args.length > 0 && args[0].equals("cluster")) {
            try {
                StormSubmitter.submitTopology("ClusterReadingFromApp", new Config(), buil
der.createTopology());
            } catch (AlreadyAliveException | InvalidTopologyException | AuthorizationExce
ption e) {
                e.printStackTrace();
            }
        } else {
            LocalCluster cluster = new LocalCluster();
            cluster.submitTopology("LocalReadingFromApp",new Config(), builder.createTopo
logy());
        }
    }
    private static SpoutConfig<String, String> getSpoutConfig(String bootstrapServers, St
ring topic) {
        return SpoutConfig.builder(bootstrapServers, topic)
                .setProp(ConsumerConfig.GROUP_ID_CONFIG, "SpoutTestGroup")
                .setRetry(getRetryService())
                .setOffsetCommitPeriodMs(10_000)
                .build();
    }
    // 定义重试策略
    private static SpoutRetryService getRetryService() {
        return new SpoutRetryExponentialBackoff(TimeInterval.microSeconds(500),
                TimeInterval.milliSeconds(2), Integer.MAX_VALUE, TimeInterval.seconds(10));

    }
}
```

```
package com.cy.ftamlirtp.common;
import java.util.ArrayList;
import java.util.List;
import cn.hutool.core.text.StrSplitter;
import cn.hutool.core.util.ArrayUtil;
import cn.hutool.core.util.StrUtil;
public class ArraySort implements Runnable {
    private int number;
    public ArraySort(int number) {
        this.number = number;
    }

    public static void main(String[] args) {
        int[] numbers = new int[]{102, 338, 62, 9132, 580, 666};
        for (int number : numbers) {
            new Thread(new ArraySort(number)).start();
        }
    }
    @Override
    public void run() {
        try {
            Thread.sleep(this.number);
            System.out.println(this.number);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
    public static List<String> parseColumn(String line) {
        line = line.replaceAll("[: :*-]", "");
        List<String> list = new ArrayList<>();
        if (!line.contains(",")) {
            list.add(line);
            list.add(StrUtil.EMPTY);
            return list;
        }
        list.add(line.substring(0, line.lastIndexOf(",")).trim());
        list.add(line.substring(line.lastIndexOf(",") + 1).trim());
        return list;

    }
    public static String parseColumnName(String str) {
        return str.substring(0, str.indexOf(" ")).trim();
    }
    public static String parseColumnType(String line) {
        List<String> words = StrSplitter.splitTrim(line, " ", true);
        if (line.contains("DECIMAL")) {
            String colType = words.get(1);
            if (!colType.contains(")")) {
                colType += words.get(2);
            }
            if (colType.endsWith(",")) {
                colType = colType.substring(0, colType.lastIndexOf(","));
            }
            return colType.trim();
        }
        return words.get(1).replace(",", "").trim();
    }
}
<?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.cy.ftamlirtp.mapper.CustomerMapper">
    <resultMap type="Customer" id="CustomerResult">
        <result property="customerId" column="customer_ID"    />
        <result property="phoneNumber" column="phone_number"    />
        <result property="updatedAt" column="updated_at"    />
        <result property="riskLevel" column="risk_level"    />
        <result property="occupation" column="occupation"    />
        <result property="name" column="name"    />
        <result property="nationality" column="nationality"    />
        <result property="status" column="status"    />
        <result property="lastReviewDate" column="last_review_date"    />
        <result property="address" column="address"    />
        <result property="nextReviewDate" column="next_review_date"    />
        <result property="onboardDate" column="onboard_date"    />
        <result property="createdAt" column="created_at"    />
        <result property="email" column="email"    />
        <result property="gender" column="gender"    />
        <result property="dateOfBirth" column="date_of_birth"    />
        <result property="customerType" column="customer_type"    />
    </resultMap>
    <sql id="selectCustomerVo">
        select customer_ID, phone_number, updated_at, risk_level, occupation, name, natio
nality, status, last_review_date, address, next_review_date, onboard_date, created_at, em
ail, gender, date_of_birth, customer_type
        from ftamlirtp_customer
    </sql>
    <select id="selectCustomerList" parameterType="Customer" resultMap="CustomerResult">
        <include refid="selectCustomerVo"/>
        <where>
            <if test="customerId != null ">
            and customer_ID = #{customerId}
            </if>
            <if test="phoneNumber != null and phoneNumber != ''">
            and phone_number = #{phoneNumber}
            </if>
            <if test="updatedAt != null ">
            and updated_at = #{updatedAt}
            </if>
            <if test="riskLevel != null and riskLevel != ''">
            and risk_level = #{riskLevel}
            </if>
            <if test="occupation != null and occupation != ''">
            and occupation = #{occupation}
            </if>
            <if test="name != null and name != ''">
            and name = #{name}
            </if>
            <if test="nationality != null and nationality != ''">
            and nationality = #{nationality}
            </if>
            <if test="status != null and status != ''">
            and status = #{status}
            </if>
            <if test="lastReviewDate != null and lastReviewDate != ''">
            and last_review_date = #{lastReviewDate}
            </if>
            <if test="address != null and address != ''">
            and address = #{address}
            </if>
            <if test="nextReviewDate != null and nextReviewDate != ''">
            and next_review_date = #{nextReviewDate}
            </if>
            <if test="onboardDate != null and onboardDate != ''">
            and onboard_date = #{onboardDate}
            </if>
            <if test="createdAt != null ">
            and created_at = #{createdAt}
            </if>
            <if test="email != null and email != ''">
            and email = #{email}
            </if>
            <if test="gender != null and gender != ''">
            and gender = #{gender}
            </if>
            <if test="dateOfBirth != null and dateOfBirth != ''">
            and date_of_birth = #{dateOfBirth}
            </if>
            <if test="customerType != null and customerType != ''">
            and customer_type = #{customerType}
            </if>
        </where>
    </select>
    <delete id="deleteCustomerByCustomerId" parameterType="Integer">
        delete from ftamlirtp_customer where customer_ID = #{customerId}
    </delete>

    <insert id="insertCustomer" parameterType="Customer" useGeneratedKeys="true" keyPrope
rty="customerId">
        insert into ftamlirtp_customer
        <trim prefix="(" suffix=")" suffixOverrides=",">
            <if test="phoneNumber != null and phoneNumber != ''">phone_number,</if>
            <if test="updatedAt != null">updated_at,</if>
            <if test="riskLevel != null and riskLevel != ''">risk_level,</if>
            <if test="occupation != null and occupation != ''">occupation,</if>
            <if test="name != null and name != ''">name,</if>
            <if test="nationality != null and nationality != ''">nationality,</if>
            <if test="status != null and status != ''">status,</if>
            <if test="lastReviewDate != null and lastReviewDate != ''">last_review_date,<
/if>
            <if test="address != null and address != ''">address,</if>
            <if test="nextReviewDate != null and nextReviewDate != ''">next_review_date,<
/if>
            <if test="onboardDate != null and onboardDate != ''">onboard_date,</if>
            <if test="createdAt != null">created_at,</if>
            <if test="email != null and email != ''">email,</if>
            <if test="gender != null and gender != ''">gender,</if>
            <if test="dateOfBirth != null and dateOfBirth != ''">date_of_birth,</if>
            <if test="customerType != null and customerType != ''">customer_type,</if>
         </trim>
        <trim prefix="values (" suffix=")" suffixOverrides=",">
            <if test="phoneNumber != null and phoneNumber != ''">#{phoneNumber},</if>
            <if test="updatedAt != null">#{updatedAt},</if>
            <if test="riskLevel != null and riskLevel != ''">#{riskLevel},</if>
            <if test="occupation != null and occupation != ''">#{occupation},</if>
            <if test="name != null and name != ''">#{name},</if>
            <if test="nationality != null and nationality != ''">#{nationality},</if>
            <if test="status != null and status != ''">#{status},</if>
            <if test="lastReviewDate != null and lastReviewDate != ''">#{lastReviewDate},
</if>
            <if test="address != null and address != ''">#{address},</if>
            <if test="nextReviewDate != null and nextReviewDate != ''">#{nextReviewDate},
</if>
            <if test="onboardDate != null and onboardDate != ''">#{onboardDate},</if>
            <if test="createdAt != null">#{createdAt},</if>
            <if test="email != null and email != ''">#{email},</if>
            <if test="gender != null and gender != ''">#{gender},</if>
            <if test="dateOfBirth != null and dateOfBirth != ''">#{dateOfBirth},</if>
            <if test="customerType != null and customerType != ''">#{customerType},</if>
         </trim>
    </insert>
    <delete id="deleteCustomerByCustomerIds" parameterType="String">
        delete from ftamlirtp_customer where customer_ID in
        <foreach item="customerId" collection="array" open="(" separator="," close=")">
            #{customerId}
        </foreach>
    </delete>
</mapper>
package com.cy.ftamlirtp.mapper;
import java.util.List;
import com.cy.ftamlirtp.domain.MonitoringRule;
import com.baomidou.mybatisplus.core.mapper.BaseMapper;
/**
 * 监控规则 Dao 接口
 */
public interface MonitoringRuleMapper extends BaseMapper<MonitoringRule>{
    /**
     * 新增监控规则
     *
     * @param monitoringRule 监控规则
     * @return
     */
    int insertMonitoringRule(MonitoringRule monitoringRule);

    /**
     * 修改监控规则
     *
     * @param monitoringRule 监控规则
     * @return
     */
    int updateMonitoringRule(MonitoringRule monitoringRule);

    /**
     * 查询监控规则列表
     *
     * @param ruleId 监控规则主键
     * @return
     */
    MonitoringRule selectMonitoringRuleByRuleId(Integer ruleId);

    /**
     * 根据条件查询监控规则
     */
    List<MonitoringRuleVo> selectByCondition(MonitoringRuleQueryDto queryMonitoringRuleDt
o);

    /**
     * 删除监控规则
     * @param ruleId 监控规则主键
     * @return
     */
    int deleteMonitoringRuleByRuleId(Integer ruleId);

    /**
     * 批量删除监控规则
     * @param ruleIds  ID 集合
     * @return
     */
    int deleteMonitoringRuleByRuleIds(Integer[] ruleIds);
}
```

```
<?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.cy.ftamlirtp.mapper.AlertRecordMapper">
    <resultMap type="AlertRecord" id="AlertRecordResult">
        <result property="alertId" column="alert_ID"    />
        <result property="severity" column="severity"    />
        <result property="alertType" column="alert_type"    />
        <result property="handled" column="handled"    />
        <result property="createdAt" column="created_at"    />
        <result property="updatedAt" column="updated_at"    />
        <result property="alertDate" column="alert_date"    />
        <result property="ruleId" column="rule_ID"    />
        <result property="description" column="description"    />
        <result property="transactionId" column="transaction_ID"    />
    </resultMap>
    <sql id="selectAlertRecordVo">
        select alert_ID, severity, alert_type, handled, created_at, updated_at, alert_dat
e, rule_ID, description, transaction_ID
        from ftamlirtp_alertrecord
    </sql>
    <select id="selectAlertRecordList" parameterType="AlertRecord" resultMap="AlertRecord
Result">
        <include refid="selectAlertRecordVo"/>
        <where>
            <if test="alertId != null ">
            and alert_ID = #{alertId}
            </if>
            <if test="severity != null ">
            and severity = #{severity}
            </if>
            <if test="alertType != null and alertType != ''">
            and alert_type = #{alertType}
            </if>
            <if test="handled != null ">
            and handled = #{handled}
            </if>
            <if test="createdAt != null ">
            and created_at = #{createdAt}
            </if>
            <if test="updatedAt != null ">
            and updated_at = #{updatedAt}
            </if>
            <if test="alertDate != null and alertDate != ''">
            and alert_date = #{alertDate}
            </if>
            <if test="ruleId != null ">
            and rule_ID = #{ruleId}
            </if>
            <if test="description != null and description != ''">
            and description = #{description}
            </if>
            <if test="transactionId != null ">
            and transaction_ID = #{transactionId}
            </if>
        </where>
    </select>
    <delete id="deleteAlertRecordByAlertId" parameterType="Integer">
        delete from ftamlirtp_alertrecord where alert_ID = #{alertId}
    </delete>

    <insert id="insertAlertRecord" parameterType="AlertRecord" useGeneratedKeys="true" ke
yProperty="alertId">
        insert into ftamlirtp_alertrecord
        <trim prefix="(" suffix=")" suffixOverrides=",">
            <if test="severity != null">severity,</if>
            <if test="alertType != null and alertType != ''">alert_type,</if>
            <if test="handled != null">handled,</if>
            <if test="createdAt != null">created_at,</if>
            <if test="updatedAt != null">updated_at,</if>
            <if test="alertDate != null and alertDate != ''">alert_date,</if>
            <if test="ruleId != null">rule_ID,</if>
            <if test="description != null and description != ''">description,</if>
            <if test="transactionId != null">transaction_ID,</if>
         </trim>
        <trim prefix="values (" suffix=")" suffixOverrides=",">
            <if test="severity != null">#{severity},</if>
            <if test="alertType != null and alertType != ''">#{alertType},</if>
            <if test="handled != null">#{handled},</if>
            <if test="createdAt != null">#{createdAt},</if>
            <if test="updatedAt != null">#{updatedAt},</if>
            <if test="alertDate != null and alertDate != ''">#{alertDate},</if>
            <if test="ruleId != null">#{ruleId},</if>
            <if test="description != null and description != ''">#{description},</if>
            <if test="transactionId != null">#{transactionId},</if>
         </trim>
    </insert>
    <delete id="deleteAlertRecordByAlertIds" parameterType="String">
        delete from ftamlirtp_alertrecord where alert_ID in
        <foreach item="alertId" collection="array" open="(" separator="," close=")">
            #{alertId}
        </foreach>
    </delete>
</mapper>
package com.cy.ftamlirtp.common.sms.util;
import cn.hutool.json.JSON;
import cn.hutool.json.JSONArray;
import cn.hutool.json.JSONObject;
import cn.hutool.json.JSONUtil;
import com.joolun.cloud.common.sms.config.SmsConfigProperties;
import lombok.AllArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.apache.http.HttpResponse;
import org.apache.http.HttpStatus;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Configuration;
import com.aliyuncs.CommonRequest;
import com.aliyuncs.CommonResponse;
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.IAcsClient;
import com.aliyuncs.exceptions.ClientException;
import com.aliyuncs.exceptions.ServerException;
import com.aliyuncs.http.MethodType;
import com.aliyuncs.profile.DefaultProfile;
import org.yaml.snakeyaml.Yaml;
import java.nio.charset.Charset;
import java.util.HashMap;
import java.util.Map;
@Slf4j
@Configuration
@AllArgsConstructor
public class SmsUtils {
    @Autowired
    private  SmsConfigProperties smsConfigProperties;
    public void sendSms(String signName, String phoneNumbers, String templateCode,
        String templateParam) throws ClientException {
        if ("1".equals(smsConfigProperties.getType())) {
            DefaultProfile profile = DefaultProfile.getProfile(smsConfigProperties.getReg
ionId(),
                smsConfigProperties.getAccessKeyId(), smsConfigProperties.getAccessKeySec
ret());
            IAcsClient client = new DefaultAcsClient(profile);
            CommonRequest request = new CommonRequest();
            request.setSysMethod(MethodType.POST);
            request.setSysDomain("dysmsapi.aliyuncs.com");
            request.setSysVersion("2017-05-25");
            request.setSysAction("SendSms");
            request.putQueryParameter("RegionId", smsConfigProperties.getRegionId());
            request.putQueryParameter("PhoneNumbers", phoneNumbers);
            request.putQueryParameter("SignName", signName);
            request.putQueryParameter("TemplateCode", templateCode);
            request.putQueryParameter("TemplateParam", "{\"code\":\""+templateParam+"\"}");

            CommonResponse response = client.getCommonResponse(request);
            String data = response.getData();
            JSONObject dataJson = JSONUtil.parseObj(data);
            if(!"OK".equals(dataJson.get("Code"))){
                log.error("发送短信失败：",data);
                throw new RuntimeException("发送短信失败：" + dataJson.get("Message"));
            }
        } else if ("2".equals(smsConfigProperties.getType())) {
            //通用短信
            Map<String, String> headerMap = new HashMap<>();
            //应用编码-测试
            headerMap.put("appCode", "message-manage-portal");
            JSONObject jsonObject = new JSONObject();
            JSONArray jsonArray=new JSONArray();
            JSONObject mobiles = new JSONObject();
            JSONObject param = new JSONObject();
            jsonObject.set("contentType", "SMS_TEXT");
            mobiles.set("name", phoneNumbers);
            mobiles.set("phone", phoneNumbers);
            jsonArray.add(mobiles);
            jsonObject.set("mobiles", jsonArray);
            jsonObject.set("templateId", "9f7cb909-b8ca-4fdd-ac0c-1d858699ffda");
            jsonObject.set("showSign", false);
            param.set("p0", templateParam);
            jsonObject.set("param", param);
            try {
                log.info("请求发送短信 url：" + smsConfigProperties.getSmsUrl());
                log.info("请求发送短信参数：" + jsonObject.toString());
                log.info("请求头发送短信：" + headerMap);
                String res = sendPostJson(smsConfigProperties.getSmsUrl(), jsonObject.toS
tring(), headerMap);
                log.info("发送短信接口返回结果" + res);
            } catch (Exception e) {
                log.info("发送短信接口失败"+e.getMessage());
                e.printStackTrace();
            }
        }
    }
    /**
     * 方法描述：发送 post 请求-json 数据
     */
    public static String sendPostJson(String url, String json, Map<String, String> header)
throws Exception {
        HttpPost httpPost = null;
        String body = "";
        try {
            CloseableHttpClient httpClient = HttpClients.createDefault();
            httpPost = new HttpPost(url);
            httpPost.setHeader("Content-type", "application/json; charset=utf-8");
            httpPost.setHeader("User-Agent", "Mozilla/4.0 (compatible; MSIE 5.0; Windows
NT; DigExt)");
```

```
            if (header != null) {
                for (Map.Entry<String, String> entry : header.entrySet()) {
                    httpPost.setHeader(entry.getKey(), entry.getValue());
                }
            }
            StringEntity entity = new StringEntity(json, Charset.forName("UTF-8"));
            entity.setContentEncoding("UTF-8");
            entity.setContentType("application/json");
            httpPost.setEntity(entity);
            HttpResponse response = httpClient.execute(httpPost);
            int statusCode = response.getStatusLine().getStatusCode();
            if (statusCode != HttpStatus.SC_OK) {
                throw new RuntimeException("请求失败");
            } else {
                body = EntityUtils.toString(response.getEntity(), "UTF-8");
            }
        } catch (Exception e) {
            throw e;
        } finally {
            if (httpPost != null) {
                httpPost.releaseConnection();
            }
        }
        return body;
    }
}
package com.cy.ftamlirtp.common.excel;
import com.alibaba.excel.metadata.Head;
import com.alibaba.excel.write.merge.AbstractMergeStrategy;
import com.baomidou.mybatisplus.core.toolkit.CollectionUtils;
import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.ss.util.CellRangeAddress;
import java.util.ArrayList;
import java.util.List;
/**
 * EasyExcel 自定义合并策略
 **/
public class CommonExcelCustStrategy extends AbstractMergeStrategy {
    /**
     * 分组，每几行合并一次
     */
    private final List<Integer> exportFieldGroupCountList;
    /**
     * 目标合并列 index
     */
    private final Integer targetColumnIndex;
    /**
     * 需要开始合并单元格的首行 index
     */
    private Integer rowIndex;

    public CommonExcelCustomMergeStrategy(List<String> exportDataList, Integer targetColu
mnIndex) {
        this.exportFieldGroupCountList = getGroupCountList(exportDataList);
        this.targetColumnIndex = targetColumnIndex;
    }

    @Override
    protected void merge(Sheet sheet, Cell cell, Head head, Integer relativeRowIndex) {

        if (null == rowIndex) {
            rowIndex = cell.getRowIndex();
        }
        // 仅从首行以及目标列的单元格开始合并，忽略其他
        if (cell.getRowIndex() == rowIndex && cell.getColumnIndex() == targetColumnIndex)
{
            mergeGroupColumn(sheet);
        }
    }

    private void mergeGroupColumn(Sheet sheet) {
        int rowCount = rowIndex;
        for (Integer count : exportFieldGroupCountList) {
            if(count == 1) {
                rowCount += count;
                continue ;
            }
            // 合并单元格
            CellRangeAddress cellRangeAddress = new CellRangeAddress(rowCount, rowCount +
count - 1, targetColumnIndex, targetColumnIndex);
            sheet.addMergedRegionUnsafe(cellRangeAddress);
            rowCount += count;
        }
    }

    private List<Integer> getGroupCountList(List<String> exportDataList){
        if (CollectionUtils.isEmpty(exportDataList)) {
            return new ArrayList<>();
        }

        List<Integer> groupCountList = new ArrayList<>();
        int count = 1;

        for (int i = 1; i < exportDataList.size(); i++) {
            if (exportDataList.get(i).equals(exportDataList.get(i - 1))) {
                count++;
            } else {
                groupCountList.add(count);
                count = 1;
            }
        }
        // 处理完最后一条后
        groupCountList.add(count);
        return groupCountList;
    }
}
package com.cy.ftamlirtp.common.util;
import java.lang.reflect.Method;
import java.net.URL;
import java.net.URLClassLoader;
public class ClassLoaderUtil {
    public static void loadJar(ClassLoader classLoader, URL jar) throws Throwable {
        if (classLoader instanceof URLClassLoader) {
            urlClassloaderLoadJar((URLClassLoader) classLoader, jar);
        } else {
            try {
                Method method = classLoader.getClass().getDeclaredMethod("appendToClassPa
thForInstrumentation", String.class);
                method.setAccessible(true);
                method.invoke(classLoader, jar.getFile());
            } catch (Throwable e) {
                e.printStackTrace();
                throw e;
            }
        }
    }

    public static ClassLoader systemLoadJar(URL jar) throws Throwable {
        ClassLoader classLoader = ClassLoader.getSystemClassLoader();
        loadJar(classLoader, jar);
        return classLoader;
    }

    private static void urlClassloaderLoadJar(URLClassLoader classLoader, URL jar) throws
Throwable {
        Method addURL = classLoader.getClass().getSuperclass().getDeclaredMethod("addURL",
URL.class);
        addURL.setAccessible(true);
        addURL.invoke(classLoader, jar);
    }
}
package com.cy.ftamlirtp.controller;
import java.util.List;
import javax.servlet.http.HttpServletResponse;
import com.cy.ftamlirtp.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.cy.ftamlirtp.common.annotation.Log;
import com.cy.ftamlirtp.common.core.controller.BaseController;
import com.cy.ftamlirtp.common.utils.poi.ExcelUtil;
import com.cy.ftamlirtp.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.cy.ftamlirtp.common.enums.BusinessType;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import com.cy.ftamlirtp.domain.MonitoringRule;
import com.cy.ftamlirtp.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.setUpdatedAt(MonitoringRuleConvUtil(i.getUpdatedAt));
            i.setDescription(MonitoringRuleConvUtil(i.getDescription));
            i.setRuleId(MonitoringRuleConvUtil(i.getRuleId));
            i.setEnabled(MonitoringRuleConvUtil(i.getEnabled));
            i.setName(MonitoringRuleConvUtil(i.getName));
        });
        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.setUpdatedAt(MonitoringRuleExportFormat(i.getUpdatedAt));
            i.setDescription(MonitoringRuleExportFormat(i.getDescription));
            i.setRuleId(MonitoringRuleExportFormat(i.getRuleId));
            i.setEnabled(MonitoringRuleExportFormat(i.getEnabled));
            i.setName(MonitoringRuleExportFormat(i.getName));
        });
        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.cy.ftamlirtp.modules.admin.controller;
import cn.hutool.core.lang.Snowflake;
import cn.hutool.core.util.IdUtil;
import com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;
import com.baomidou.mybatisplus.core.conditions.update.LambdaUpdateWrapper;
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;
import com.cy.ftamlirtp.common.ErrorCodeEnum;
import com.cy.ftamlirtp.common.annotation.SysLog;
import com.cy.ftamlirtp.common.constant.SystemConstant;
import com.cy.ftamlirtp.common.utils.result.Rsp;
import com.cy.ftamlirtp.common.utils.result.RunLog;
import com.cy.ftamlirtp.modules.admin.entity.SysRoleEntity;
import com.cy.ftamlirtp.modules.admin.entity.SysUserEntity;
import com.cy.ftamlirtp.modules.admin.entity.SysUserRoleEntity;
import com.cy.ftamlirtp.modules.admin.service.SysRoleServiceImpl;
import com.cy.ftamlirtp.modules.admin.service.SysUserRoleServiceImpl;
import com.cy.ftamlirtp.modules.admin.service.SysUserServiceImpl;
import com.cy.ftamlirtp.modules.admin.vo.SysUserRoleVo;
import com.cy.ftamlirtp.modules.admin.vo.user.SysUserFrozenReqVo;
import com.cy.ftamlirtp.modules.admin.vo.user.SysUserVo;
import com.cy.ftamlirtp.modules.admin.vo.user.UpdateUserPasswordVo;
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.apache.commons.lang3.StringUtils;
import org.springframework.beans.BeanUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.web.bind.annotation.GetMapping;
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 javax.servlet.http.HttpServletResponse;
import javax.validation.Valid;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Date;
import java.util.List;
import java.util.stream.Collectors;
@Slf4j
@RestController
@RequestMapping("/sys/user")
@Api(tags = "系统用户")
public class SysUserController {

    @Autowired
    private SysUserServiceImpl sysUserService;
    @Autowired
    private SysUserRoleServiceImpl userRoleService;
    @Autowired
    private SysRoleServiceImpl sysRoleService;

    @SysLog(value = "新增用户", isSaveDatabase = true)
    @ApiOperation(value = "新增用户")
    @RequestMapping(value = "/add", method = RequestMethod.POST)
    public Rsp add(@RequestBody SysUserVo userReq) {
        String selectedRoles = userReq.getSelectedRoleId(); // 用户角色
        String selectedSystemCodes = userReq.getSelectedSystemCode();  // 用户所属子系统
编码
        try {
            SysUserEntity user = new SysUserEntity();
            BeanUtils.copyProperties(userReq, user);
            sysUserService.saveUser(user, selectedRoles, selectedSystemCodes);
            return Rsp.success();
        } catch (Exception e) {
            log.error(e.getMessage(), e.fillInStackTrace());
            return Rsp.error(e.getMessage());
        }
    }

    @SysLog(value = "修改用户", isSaveDatabase = true)
    @ApiOperation(value = "修改用户")
    @RequestMapping(value = "/edit", method = RequestMethod.PUT)
    public Rsp edit(@RequestBody SysUserVo userReq) {
        try {
            Long userId = userReq.getId();
            SysUserEntity sysUser = sysUserService.getById(userId);
            if (sysUser == null) {
                return Rsp.error("当前用户不存在");
            }
            if (sysUser.getSysDefault() == SystemConstant.SysDefaultEnum.DEFAULT_0.getCod
e()) {
                return Rsp.error("系统用户，无法修改");
            }
            sysUserService.editUser(userReq);
            return Rsp.success();
        } catch (Exception e) {
            log.error(e.getMessage(), e.fillInStackTrace());
            return Rsp.error(e.getMessage());
        }
    }

    @SysLog(value = "删除用户", isSaveDatabase = true, readRunLog = true)
    @ApiOperation(value = "删除用户")
    @RequestMapping(value = "/delete", method = RequestMethod.DELETE)
    public Rsp delete(@RequestParam(name = "id") Long id) {
        try {
            SysUserEntity user = sysUserService.getById(id);
            if (user == null) {
                return Rsp.error("删除用户不存在");
            }
            if (user.getSysDefault() == SystemConstant.SysDefaultEnum.DEFAULT_0.getCode())
{
                return Rsp.error("系统用户，不能删除");
            }
            this.sysUserService.deleteUser(id);
            return Rsp.successWithLog(RunLog.create(user));
        } catch (Exception e) {
            log.error(e.getMessage(), e.fillInStackTrace());
            return Rsp.error(e.getMessage());
        }
    }

    @SysLog(value = "批量删除用户", isSaveDatabase = true, readRunLog = true)
    @ApiOperation(value = "批量删除用户")
    @RequestMapping(value = "/deleteBatch", method = RequestMethod.DELETE)
    public Rsp deleteBatch(@RequestParam(name = "ids") String ids) {
        List<Long> idList = Arrays.stream(ids.split(",")).map(Long::parseLong).collect(Co
llectors.toList());
        List<SysUserEntity> list = sysUserService.list(new LambdaQueryWrapper<SysUserEnti
ty>()
                .select(SysUserEntity::getId, SysUserEntity::getUsername).in(SysUserEntit
y::getId, idList));
        if (list != null && list.size() > 0) {
            this.sysUserService.deleteBatchUsers(list.stream().map(SysUserEntity::getId).
collect(Collectors.toList()));
        }
        return Rsp.successWithLog(RunLog.create(list));
    }

    @SysLog(value = "冻结&解冻用户", isSaveDatabase = true, readRunLog = true)
    @ApiOperation(value = "冻结&解冻用户")
    @RequestMapping(value = "/frozenBatch", method = RequestMethod.PUT)
    public Rsp frozenBatch(@Valid @RequestBody SysUserFrozenReqVo reqVo) {
        try {
            List<Long> ids = reqVo.getIds();
            sysUserService.frozenUserBatch(ids, reqVo.getStatus());
            return Rsp.successWithLog(RunLog.create(new ArrayList<>(sysUserService.listBy
Ids(ids))));
        } catch (Exception e) {
            log.error(e.getMessage(), e.fillInStackTrace());
            return Rsp.error("修改失败");
        }
    }

    @ApiOperation(value = "查询用户列表")
    @ApiImplicitParams({
            @ApiImplicitParam(paramType = "query", name = "pageNo", value = "当前页码", d
ataType = "Long"),
            @ApiImplicitParam(paramType = "query", name = "pageSize", value = "每页显示记
录数", dataType = "Long")
    })
    @GetMapping(value = "/list")
    @SysLog(value = "查询用户列表", isSaveDatabase = true)
    public Rsp list(SysUserVo user, @RequestParam(name = "pageNo", defaultValue = "1") In
teger pageNo,
        @RequestParam(name = "pageSize", defaultValue = "10") Integer pageSize) {
        return Rsp.success(sysUserService.queryPage(user, pageNo, pageSize));
    }
    @ApiOperation(value = "查询可用的自定义用户信息")
    @ApiImplicitParams({
            @ApiImplicitParam(paramType = "query", name = "pageNo", value = "当前页码", d
ataType = "int"),
            @ApiImplicitParam(paramType = "query", name = "pageSize", value = "每页显示记
录数", dataType = "int")
    })
    @GetMapping(value = "/available_custom_user_list")
    @SysLog(value = "查询可用的自定义用户信息", isSaveDatabase = true)
    public Rsp availableCustomUserList(SysUserVo user, @RequestParam(name = "pageNo", def
aultValue = "1") Integer pageNo,
        @RequestParam(name = "pageSize", defaultValue = "10") Integer pageSize) {
        return Rsp.success(sysUserService.queryAvailableCustomUserPage(user, pageNo, page
Size));
    }

    @SysLog(value = "查询单个用户信息", isSaveDatabase = true)
    @ApiOperation(value = "查询单个用户", notes = "根据用户 id 查询单个用户")
    @RequestMapping(value = "/queryById", method = RequestMethod.GET)
    public Rsp queryById(@RequestParam(name = "id") Long id) {
        SysUserEntity sysUser = sysUserService.getById(id);
        if (sysUser == null) {
            return Rsp.error("用户不存在");
        } else {
            return Rsp.success(sysUser);
        }
    }

    @SysLog(value = "查询用户的的角色 ID", isSaveDatabase = true)
    @ApiOperation(value = "查询用户的的角色 ID", notes = "查询用户的的角色 ID")
    @RequestMapping(value = "/queryUserRole", method = RequestMethod.GET)
    public Rsp queryUserRole(@RequestParam(name = "userid") Long userid, @RequestParam(na
me = "systemCode") String systemCode) {
        List<Long> roleId = sysRoleService.list(new LambdaQueryWrapper<SysRoleEntity>()
            .eq(SysRoleEntity::getSystemCode, systemCode)).stream().map(SysRoleEntity::ge
tId).collect(Collectors.toList());
        List<Long> userRoleId = userRoleService.list(new LambdaQueryWrapper<SysUserRoleEn
tity>()
            .eq(SysUserRoleEntity::getUserId, userid).in(SysUserRoleEntity::getRoleId, ro
leId))
                .stream().map(SysUserRoleEntity::getRoleId).collect(Collectors.toList());
        return Rsp.success(userRoleId);
    }

    @SysLog(value = "获取同一角色下的用户列表", isSaveDatabase = true)
    @ApiOperation(value = "获取同一角色下的用户列表")
    @RequestMapping(value = "/userRoleList", method = RequestMethod.GET)
    public Rsp userRoleList(
            @RequestParam(name = "roleId") Long roleId,
            @RequestParam(name = "username", required = false) String username,
            @RequestParam(name = "pageNo", defaultValue = "1") Integer pageNo,
            @RequestParam(name = "pageSize", defaultValue = "10") Integer pageSize) {
        Page<SysUserEntity> page = new Page<>(pageNo, pageSize);
        return Rsp.success(sysUserService.getUserByRoleId(page, roleId, username));
    }

    @SysLog(value = "修改密码", isSaveDatabase = true)
    @ApiOperation(value = "修改密码")
    @RequestMapping(value = "/changePassword", method = RequestMethod.PUT)
    public Rsp changePassword(@RequestBody UpdateUserPasswordVo updateUserPasswordVo) {
        SysUserEntity u = sysUserService.getUserByName(updateUserPasswordVo.getUsername(),
updateUserPasswordVo.getSystemCode());
        if (u == null) {
            return Rsp.success(ErrorCodeEnum.FAILED.getCode(), "用户不存在！");
        }
        // 判断旧密码是否正确
        if (StringUtils.isNotBlank(updateUserPasswordVo.getOldPassword()) &&
                !new BCryptPasswordEncoder().matches(updateUserPasswordVo.getOldPassword(),
 u.getPassword())) {
            return Rsp.success(ErrorCodeEnum.FAILED.getCode(), "旧密码不正确");
        }
        // 判断用户历史密码和新密码是否相同
        if (new BCryptPasswordEncoder().matches(updateUserPasswordVo.getPassword(), u.get
Password())) {
            return Rsp.success(ErrorCodeEnum.FAILED.getCode(), "旧密码和新密码相同");
        }
        // 修改新密码
        String newPassword = new BCryptPasswordEncoder().encode(updateUserPasswordVo.getP
assword());
        boolean update = sysUserService.update(new LambdaUpdateWrapper<SysUserEntity>()
                .set(SysUserEntity::getPassword, newPassword).set(SysUserEntity::getIsFir
stLogin, 0)
                .set(SysUserEntity::getModifyPasswordTime, new Date()).eq(SysUserEntity::
getId, u.getId()));
        if (!update) {
            return Rsp.success(ErrorCodeEnum.FAILED.getCode(), "修改密码失败");
        }
        return Rsp.success();
    }


    @SysLog(value = "用户下拉框选择", isSaveDatabase = true)
    @ApiOperation(value = "用户下拉框选择")
    @RequestMapping(value = "/select_user_list", method = RequestMethod.GET)
    public Rsp getUserDepartsList(@RequestParam(name = "systemCode") String systemCode) {
        return Rsp.success(sysUserService.selectUserList(systemCode));
    }

    @SysLog(value = "生成用户 id")
    @ApiOperation(value = "生成用户 id", notes = "生成在添加用户情况下没有主键的问题,返回
给前端,根据该 id 绑定部门数据")
    @RequestMapping(value = "/generateUserId", method = RequestMethod.GET)
    public Rsp generateUserId() {
        Snowflake snowflake = IdUtil.getSnowflake();
        return Rsp.success(snowflake.nextIdStr());
    }



    @SysLog(value = "增加角色和用户的关联", isSaveDatabase = true)
    @ApiOperation(value = "增加角色和用户的关联")
    @RequestMapping(value = "/addSysUserRole", method = RequestMethod.POST)
    public Rsp addSysUserRole(@RequestBody SysUserRoleVo sysUserRoleVo) {
        try {
            sysUserService.userRoleBind(sysUserRoleVo);
            return Rsp.success();
        } catch (Exception e) {
            log.error(e.getMessage(), e.fillInStackTrace());
            return Rsp.error(e.getMessage());
        }
    }

    @SysLog(value = "解除角色和用户的关联", isSaveDatabase = true)
    @ApiOperation(value = "解除角色和用户的关联")
    @RequestMapping(value = "/deleteUserRole", method = RequestMethod.DELETE)
    public Rsp deleteUserRole(@RequestParam(name = "roleId") Long roleId, @RequestParam(n
ame = "userId") Long userId) {
        try {
            userRoleService.remove(new LambdaQueryWrapper<SysUserRoleEntity>().
                    eq(SysUserRoleEntity::getUserId, userId).eq(SysUserRoleEntity::getRol
eId, roleId));
            return Rsp.success();
        } catch (Exception e) {
            log.error(e.getMessage(), e.fillInStackTrace());
            return Rsp.error("解除关联失败！");
        }
    }

    @SysLog(value = "批量解除角色和用户的关联", isSaveDatabase = true)
    @ApiOperation(value = "批量解除角色和用户的关联")
    @RequestMapping(value = "/deleteUserRoleBatch", method = RequestMethod.DELETE)
    public Rsp deleteUserRoleBatch(@RequestParam(name = "roleId") Long roleId, @RequestPa
ram(name = "userIds") String userIds) {
        try {
            List<Long> userIdList = Arrays.stream(userIds.split(",")).map(Long::parseLong).
collect(Collectors.toList());
            userRoleService.remove(new LambdaQueryWrapper<SysUserRoleEntity>().eq(SysUser
RoleEntity::getRoleId, roleId)
                    .in(SysUserRoleEntity::getUserId, userIdList));
            return Rsp.success();
        } catch (Exception e) {
            log.error(e.getMessage(), e.fillInStackTrace());
            return Rsp.error("删除失败！");
        }
    }


    @GetMapping("/export")
    @ApiOperation("用户导出")
    @SysLog("用户导出")
    public void exportAppAccounts(HttpServletResponse response, SysUserVo user) {
        sysUserService.exportUser(response, user);
    }
}
package com.cy.ftamlirtp.domain;
import java.math.BigDecimal;
import java.util.Date;
import com.fasterxml.jackson.annotation.JsonFormat;
import lombok.Data;
import com.cy.ftamlirtp.common.annotation.Excel;
import com.baomidou.mybatisplus.annotation.TableName;
import com.cy.ftamlirtp.common.core.domain.BaseEntity;
/**
 * 账户信息实体类
 */
@TableName("ftamlirtp_account")
@Data
public class Account extends BaseEntity {
    private static final long serialVersionUID = --4302412779239586396LL;
    // 账户 ID
    private Integer accountId;
    // 账户余额
    private BigDecimal balance;
    // 更新时间
    private Date updatedAt;
    // 关闭日期
    private String closureDate;
    // 开立日期
    private String openingDate;
    // 账户类型
    private String accountType;
    // 账户号码
    private String accountNumber;
    // 账户状态
    private String accountStatus;
    // 创建时间
    private Date createdAt;
    // 客户 ID
    private Integer customerId;
    // 冻结状态
    private Integer freezeStatus;
    // 账户限额
    private BigDecimal limit;
    // 账户货币
    private String currency;
}
package com.cy.ftamlirtp.mapper;
import java.util.List;
import com.cy.ftamlirtp.domain.Transaction;
import com.baomidou.mybatisplus.core.mapper.BaseMapper;
/**
 * 交易记录 Dao 接口
 */
public interface TransactionMapper extends BaseMapper<Transaction>{
    /**
     * 新增交易记录
     *
     * @param transaction 交易记录
     * @return
     */
    int insertTransaction(Transaction transaction);

    /**
     * 修改交易记录
     *
     * @param transaction 交易记录
     * @return
     */
    int updateTransaction(Transaction transaction);

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

    /**
     * 根据条件查询交易记录
     */
    List<TransactionVo> selectByCondition(TransactionQueryDto queryTransactionDto);

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

    /**
```

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

package com.cy.ftamlirtp.mapper;
import java.util.List;
import com.cy.ftamlirtp.domain.Report;
import com.baomidou.mybatisplus.core.mapper.BaseMapper;
/**
 * 报告 Dao 接口
 */
public interface ReportMapper extends BaseMapper<Report>{
    /**
     * 新增报告
     *
     * @param report 报告
     * @return
     */
    int insertReport(Report report);

    /**
     * 修改报告
     *
     * @param report 报告
     * @return
     */
    int updateReport(Report report);

    /**
     * 查询报告列表
     *
     * @param reportId 报告主键
     * @return
     */
    Report selectReportByReportId(Integer reportId);

    /**
     * 根据条件查询报告
     */
    List<ReportVo> selectByCondition(ReportQueryDto queryReportDto);

    /**
     * 删除报告
     * @param reportId 报告主键
     * @return
     */
    int deleteReportByReportId(Integer reportId);
    /**
     * 批量删除报告
     * @param reportIds  ID 集合
     * @return
     */
    int deleteReportByReportIds(Integer[] reportIds);
}

package com.cy.ftamlirtp.modules.admin.service;
import cn.hutool.core.util.IdUtil;
import cn.hutool.crypto.digest.DigestUtil;
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.core.metadata.IPage;
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;
import com.cy.ftamlirtp.common.ErrorCodeEnum;
import com.cy.ftamlirtp.common.constant.SystemConstant;
import com.cy.ftamlirtp.common.exception.GlobalException;
import com.cy.ftamlirtp.common.utils.DsHttpClient;
import com.cy.ftamlirtp.common.utils.ExcelUtils;
import com.cy.ftamlirtp.common.utils.OAuth2UserUtils;
import com.cy.ftamlirtp.modules.admin.dao.SysUserDao;
import com.cy.ftamlirtp.modules.admin.entity.SysRoleEntity;
import com.cy.ftamlirtp.modules.admin.entity.SysUserBusinessEntity;
import com.cy.ftamlirtp.modules.admin.entity.SysUserEntity;
import com.cy.ftamlirtp.modules.admin.entity.SysUserRoleEntity;
import com.cy.ftamlirtp.modules.admin.utils.JwtUtils;
import com.cy.ftamlirtp.modules.admin.vo.SysUserRoleVo;
import com.cy.ftamlirtp.modules.admin.vo.login.LoginBigDataReqVo;
import com.cy.ftamlirtp.modules.admin.vo.user.SysUserVo;
import com.cy.ftamlirtp.modules.statics.vo.DataModelEntity;
import com.google.code.kaptcha.Producer;
import lombok.extern.slf4j.Slf4j;
import org.apache.commons.lang3.StringUtils;
import org.springframework.beans.BeanUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.data.redis.core.StringRedisTemplate;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import javax.servlet.http.HttpServletResponse;
import java.awt.image.BufferedImage;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.concurrent.TimeUnit;
import java.util.stream.Collectors;
@Slf4j
@Service("sysUserService")
public class SysUserServiceImpl extends ServiceImpl<SysUserDao, SysUserEntity> {
    @Autowired
    private SysUserRoleServiceImpl sysUserRoleService;
    @Autowired
    private SysUserBusinessServiceImpl sysUserBusinessService;
    @Autowired
    private SysDepartmentServiceImpl sysDepartService;
    @Autowired
    private StringRedisTemplate redisTemplate;
    @Autowired
    private SysRoleServiceImpl sysRoleService;
    @Autowired
    private JwtUtils jwtUtils;
    @Autowired
    private Producer producer;
    @Value("${system.token.expire}")
    private Long expireTime;
    @Value("${dfm-api.bigdataCenter.endpoint}")
    private String loginHost;
    @Value("${dfm-api.bigdataCenter.clientId}")
    private String clientId;
    @Value("${dfm-api.bigdataCenter.clientSecret}")
    private String clientSecret;

    /**
     * 分页查询用户列表
     *
     * @param user     条件查询
     * @param pageNo   页数
     * @param pageSize 页面大小
     */
    public IPage<SysUserEntity> queryPage(SysUserVo user, Integer pageNo, Integer pageSiz
e) {
        Page<SysUserEntity> page = new Page<>(pageNo, pageSize);
        if (!SystemConstant.SUPER_ADMIN.equals(OAuth2UserUtils.getOauthUserRoleCode())) {
            Long departmentId = OAuth2UserUtils.getDepartmentAuth(null);
            List<Long> departmentList = sysDepartService.findChildDepartmentIdByCurrentId
(Collections.singletonList(departmentId), true);
            user.setDepartmentList(departmentList);
        }
        IPage<SysUserEntity> result = this.baseMapper.queryUserListPage(page, user);
        result.getRecords().forEach(m -> m.setDepartmentName(sysDepartService.searchDepar
tPath(m.getDepartmentId())));
        return result;
    }

    /**
     * 查询未分配角色的用户
     *
     * @param user     用户对象参数
     * @param pageNo   第几页
     * @param pageSize 每页个数
     * @return IPage<SysUserEntity>
     */
    public IPage<SysUserEntity> queryAvailableCustomUserPage(SysUserVo user, Integer page
No, Integer pageSize) {
        Page<SysUserEntity> page = new Page<>(pageNo, pageSize);
        user.setSysDefault(1);
        user.setSelectedSystemCode(user.getSystemCode());
        if (StringUtils.isNotBlank(user.getRoleId())) {
            List<Long> tempUserId = sysUserRoleService.list(new LambdaQueryWrapper<SysUse
rRoleEntity>()
                    .in(SysUserRoleEntity::getRoleId, Arrays.stream(user.getRoleId().spli
t(",")).map(Long::parseLong).collect(Collectors.toList())))
                    .stream().map(SysUserRoleEntity::getUserId).collect(Collectors.toList
());
            if (tempUserId.size() > 0) {
                user.setExcludeUserIdList(tempUserId);
            }
        }
        if (!SystemConstant.SUPER_ADMIN.equals(OAuth2UserUtils.getOauthUserRoleCode())) {
            Long departmentId = OAuth2UserUtils.getDepartmentAuth(null);
            List<Long> departmentList = sysDepartService.findChildDepartmentIdByCurrentId
(Collections.singletonList(departmentId), true);
            user.setDepartmentList(departmentList);
        }
        IPage<SysUserEntity> result = this.baseMapper.queryUserListPage(page, user);
        result.getRecords().forEach(m -> m.setDepartmentName(sysDepartService.searchDepar
tPath(m.getDepartmentId())));
        return result;
    }

    /**
     * 根据用户名称查询用户
     */
    public SysUserEntity getUserByName(String username, String systemCode) {
        return this.baseMapper.queryUserByUsername(username, systemCode);
    }

    /**
     * 新增用户
     *
     * @param user              用户对象参数
     * @param selectedRoles     用户角色
     * @param selectSystemCodes 用户归属业务系统
     */
    @Transactional
    public void saveUser(SysUserEntity user, String selectedRoles, String selectSystemCod
es) {
        String passwordEncode = new BCryptPasswordEncoder().encode(user.getPassword());
        user.setPassword(passwordEncode);
        user.setModifyPasswordTime(new Date());
        if (user.getId() == null) {
            user.setId(IdUtil.getSnowflake().nextId());
        }
        // step.1 保存用户  取登录用户信息，为创建人
        if (StringUtils.isBlank(user.getCreateBy())) {
            user.setCreateBy(OAuth2UserUtils.getOauthUserName());
        }
        this.baseMapper.insert(user);
        // step.2 保存用户角色
        if (StringUtils.isNotEmpty(selectedRoles)) {
            Arrays.stream(selectedRoles.split(",")).forEach(id -> sysUserRoleService.getB
aseMapper().insert(new SysUserRoleEntity(user.getId(), Long.parseLong(id))));
        }
        // step.3 保存用户所属的子系统
        if (StringUtils.isNotBlank(selectSystemCodes)) {
            Arrays.stream(selectSystemCodes.split(",")).forEach(code -> sysUserBusinessSe
rvice.getBaseMapper().insert(new SysUserBusinessEntity(user.getId(), code)));
        }

    }

    /**
     * 编辑用户信息
     *
     * @param userReq 用户对象参数
     */
    @Transactional
    public void editUser(SysUserVo userReq) {
        SysUserEntity user = new SysUserEntity();
        BeanUtils.copyProperties(userReq, user);
        if (StringUtils.isNotBlank(user.getPassword())) {
            user.setPassword(new BCryptPasswordEncoder().encode(user.getPassword()));
            user.setModifyPasswordTime(new Date());  // 设置密码修改时间
        } else {
            user.setPassword(null); // 防止传空密码被覆盖
        }
        String roles = userReq.getSelectedRoleId();
        String systemCodes = userReq.getSelectedSystemCode();
        // step.1 修改用户基础信息
        this.baseMapper.updateById(user);
        // step.2 修改角色
        if (StringUtils.isNotEmpty(roles)) {
            List<SysUserRoleEntity> relation = new ArrayList<>();
            Arrays.stream(roles.split(",")).forEach(id -> relation.add(new SysUserRoleEnt
ity(user.getId(), Long.parseLong(id))));
            // 处理用户角色 先删后加
            List<String> systemCode = sysRoleService.listByIds(relation.stream().map(SysU
serRoleEntity::getRoleId).collect(Collectors.toList())).stream().map(SysRoleEntity::getSy
stemCode).collect(Collectors.toList());
            List<Long> roleList = sysRoleService.list(new LambdaQueryWrapper<SysRoleEntit
y>().in(SysRoleEntity::getSystemCode, systemCode)).stream().map(SysRoleEntity::getId).col
lect(Collectors.toList());
            sysUserRoleService.remove(new LambdaQueryWrapper<SysUserRoleEntity>().eq(SysU
serRoleEntity::getUserId, user.getId()).in(SysUserRoleEntity::getRoleId, roleList));
            sysUserRoleService.saveBatch(relation);  // 增加角色
        }
        // step.3 修改用户所属子系统 处理用户子系统 先删后加
        sysUserBusinessService.remove(new LambdaQueryWrapper<SysUserBusinessEntity>().eq(
SysUserBusinessEntity::getUserId, user.getId()));
        if (StringUtils.isNotBlank(systemCodes)) {
            Arrays.stream(systemCodes.split(",")).forEach(code -> sysUserBusinessService.
getBaseMapper().insert(new SysUserBusinessEntity(user.getId(), code)));
        }
    }

    /**
     * 删除用户信息
     *
     * @param userId 用户 ID
     */
    @Transactional
    public void deleteUser(Long userId) {
        // 1.删除用户
        this.baseMapper.deleteById(userId);
        // 2.删除用户和角色的对应关系
        sysUserRoleService.remove(new LambdaQueryWrapper<SysUserRoleEntity>().eq(SysUserR
oleEntity::getUserId, userId));
        // 3.删除用户和子系统的对应关系
        sysUserBusinessService.remove(new LambdaQueryWrapper<SysUserBusinessEntity>().eq(
SysUserBusinessEntity::getUserId, userId));
    }

    /**
     * 批评删除用户
     *
     * @param idList 用户 ID 集合
     */
    @Transactional(rollbackFor = Exception.class)
    public void deleteBatchUsers(List<Long> idList) {
        // 0.如果批量删除中包含系统默认用户，那么删除失败
        List<SysUserEntity> users = this.list(new LambdaQueryWrapper<SysUserEntity>().in(
SysUserEntity::getId, idList).eq(SysUserEntity::getSysDefault, SystemConstant.SysDefaultE
num.DEFAULT_0.getCode()));
        if (users.size() > 0) {
            throw new GlobalException("包含系统默认用户，无法删除");
        }
        // 1.删除用户
        this.baseMapper.deleteBatchIds(idList);
        // 2. 删除用户角色关系
        sysUserRoleService.remove(new LambdaQueryWrapper<SysUserRoleEntity>().eq(SysUserR
oleEntity::getUserId, idList));
        // 3.删除用户和子系统的对应关系
        sysUserBusinessService.remove(new LambdaQueryWrapper<SysUserBusinessEntity>().eq(
SysUserBusinessEntity::getUserId, idList));
    }

    /**
     * 根据角色 Id 查询用户列表
     */
    public IPage<SysUserEntity> getUserByRoleId(Page<SysUserEntity> page, Long roleId, St
ring username) {
        IPage<SysUserEntity> result = this.baseMapper.getUserByRoleId(page, roleId, usern
ame);
        result.getRecords().forEach(m -> m.setDepartmentName(sysDepartService.searchDepar
tPath(m.getDepartmentId())));
        return result;
    }

    /**
     * 冻结用户或解除冻结
     *
     * @param ids    用户 ID 集
     * @param status 1：正常  2：冻结
     */
    @Transactional(rollbackFor = Exception.class)
    public void frozenUserBatch(List<Long> ids, Integer status) {
        this.update(new LambdaUpdateWrapper<SysUserEntity>().in(SysUserEntity::getId, ids).
set(SysUserEntity::getIsNormal, status));
    }

    /**
     * 保存用户与角色的关联信息
     *
     * @param sysUserRoleVo 关系对象信息
     */
    @Transactional(rollbackFor = Exception.class)
    public void userRoleBind(SysUserRoleVo sysUserRoleVo) {
        // 1.获取角色 id
        Long sysRoleId = sysUserRoleVo.getRoleId();
        // 2.遍历用户列表，将用户和角色建立关联（已经有关联的不再关联）
        for (Long sysUserId : sysUserRoleVo.getUserIdList()) {
            // 删除已有的绑定关系，添加新的关联
            sysUserRoleService.remove(new LambdaQueryWrapper<SysUserRoleEntity>().eq(SysU
serRoleEntity::getUserId, sysUserId));
            // 增加新关系
            sysUserRoleService.save(new SysUserRoleEntity(sysUserId, sysRoleId));
        }
    }

    /**
     * 查询用户下拉列表
     *
     * @param systemCode 系统编码
     * @return List<DataModelEntity>
     */
    public List<DataModelEntity> selectUserList(String systemCode) {
        return this.baseMapper.selectUserListDao(systemCode);
    }

    /**
     * 导出用户信息
     *
     * @param response 响应体
     * @param user     用户检索信息
     */
    public void exportUser(HttpServletResponse response, SysUserVo user) {
        IPage<SysUserEntity> dataList = this.queryPage(user, 0, -1);
        try {
            ExcelUtils.downloadExcel(response, dataList.getRecords(), SysUserEntity.class,
"用户账号信息");
        } catch (IOException e) {
            log.error(e.getMessage(), e.fillInStackTrace());
        }
    }


    public SysUserEntity getUserById(long userId) {
        SysUserEntity user = this.getById(userId);
        user.setPassword(null);
        return user;
    }

    /**
     * 生成验证码并缓存
     *
     * @param uuid 唯一标识符
     * @return BufferedImage
     */
    public BufferedImage getCaptcha(String uuid) {
        if (StringUtils.isBlank(uuid)) {
            throw new GlobalException("uuid 不能为空");
        }
        //生成文字验证码
        String code = producer.createText();
        // 将验证码放入 redis 中,有效期 5 分钟
        redisTemplate.opsForValue().set(SystemConstant.LOGIN_CAPTCHA_REDIS_KEY + uuid, co
de, 5, TimeUnit.MINUTES);
        return producer.createImage(code);
    }

    /**
     * 检查输入的验证码与缓存的是否相同
     *
     * @param uuid    唯一标识符
     * @param captcha 验证码
     * @return Boolean
     */
    public Boolean validateCaptcha(String uuid, String captcha) {
        String code = redisTemplate.opsForValue().get(SystemConstant.LOGIN_CAPTCHA_REDIS_
KEY + uuid);
        if (StringUtils.isNotBlank(code)) {
            if (code.equalsIgnoreCase(captcha)) {
                // 删除 redis 中的验证码
                redisTemplate.delete(SystemConstant.LOGIN_CAPTCHA_REDIS_KEY + uuid);
                return true;
            } else {
                return false;
            }
        } else {
            throw new GlobalException(ErrorCodeEnum.ADMIN_USER_CODE_EXPIRED);
        }
    }

    /**
     * 返回用户被锁定的剩余时间
     *
     * @param uuid 唯一标识符
     * @return Long
     */
    public Long limitLoginTime(String uuid) {
        String time = redisTemplate.opsForValue().get(SystemConstant.LOGIN_LIMIT_REDIS_KE
Y + uuid);
        if (StringUtils.isNotBlank(time)) {
            return Long.parseLong(time) - System.currentTimeMillis() / 1000;
        }
        return 0L;
    }

    /**
     * 返回用户登录失败后剩余的登录次数
     *
     * @param uuid 唯一标识符
     * @return Integer
     */
    public Integer loginFailTime(String uuid, Integer limitCount, Integer limitTime) {
        String failCount = redisTemplate.opsForValue().get(SystemConstant.LOGIN_FAIL_REDI
S_KEY + uuid);
        int lastCount;
        if (StringUtils.isBlank(failCount)) {
            redisTemplate.opsForValue().set(SystemConstant.LOGIN_FAIL_REDIS_KEY + uuid, (
limitCount - 1) + "");
            lastCount = limitCount - 1;
        } else {
            lastCount = Integer.parseInt(failCount) - 1;
        }
        if (lastCount > 0) {
            redisTemplate.opsForValue().set(SystemConstant.LOGIN_FAIL_REDIS_KEY + uuid, l
astCount + "");
            return lastCount;
        }
        long wailTime = System.currentTimeMillis() / 1000 + limitTime;
        redisTemplate.opsForValue().set(SystemConstant.LOGIN_LIMIT_REDIS_KEY + uuid, wail
Time + "", limitTime, TimeUnit.SECONDS);
        redisTemplate.delete(SystemConstant.LOGIN_FAIL_REDIS_KEY + uuid);
        return 0;
    }

    /**
     * 清除上次登录后缓存的信息
     *
     * @param uuid 唯一标识符
     */
    public void clearLoginRecord(String uuid) {
        redisTemplate.delete(SystemConstant.LOGIN_FAIL_REDIS_KEY + uuid);
        redisTemplate.delete(SystemConstant.LOGIN_LIMIT_REDIS_KEY + uuid);
    }

    /**
     * 用户登录业务处理
     */
    public Map<String, Object> login(SysUserEntity user, String systemCode) {
        Map<String, Object> map = new HashMap<>();
        String token = jwtUtils.generateToken(user.getId());
        // 将 token 放入 redis 中
        redisTemplate.opsForValue().set(SystemConstant.USER_TOKEN_REDIS_KEY + token, user.g
etId() + "#" + systemCode + "#" + user.getRoleCode(), expireTime, TimeUnit.SECONDS);
        map.put("access_token", token);
        map.put("expire", expireTime);
        map.put("systemCode", systemCode);
        map.put("userInfo", new SysUserEntity(user.getId(), user.getUsername(), user.getR
ealname(), user.getDepartmentId(), user.getSysDefault(), user.getRoleCode()));
        return map;
    }

    /**
     * 取消首次登录状态
     *
     * @param userId 用户 ID
     */
    public void modifyFirstLogin(Long userId) {
        this.update(new LambdaUpdateWrapper<SysUserEntity>().set(SysUserEntity::getIsFirs
tLogin, 0).eq(SysUserEntity::getId, userId));
    }



    /**
     * 用户退出
     */
    public void logout() {
        redisTemplate.delete(SystemConstant.USER_TOKEN_REDIS_KEY + OAuth2UserUtils.getOau
thToken());
    }

    /**
     * 大数据中心登录
     *
     * @param login 登录信息
     * @return Map<String, Object>
     */
    public Map<String, Object> loginBigdataCenter(LoginBigDataReqVo login) {
        String token_url = loginHost + "/authn/token";
        String person_url = loginHost + "/id/person?access_token=";
        Map<String, Object> param = new HashMap<>();
        param.put("grant_type", "authorization_code");
        param.put("code", login.getCode());
        param.put("redirect_uri", login.getRedirectUrl());
        String data = DsHttpClient.loginBigdataCenter(token_url, param, clientId, clientS
ecret);
        log.info("调用获取 token 接口返回数据：{}", data);
        if (StringUtils.isBlank(data) || JSONObject.parseObject(data).containsKey("error"))
 {
            throw new GlobalException("统一认证服务获取 token 异常");
        }
        JSONObject tokenInfo = JSONObject.parseObject(data);
        String token = tokenInfo.getString("access_token");
        Long time = tokenInfo.getLong("expires_in");
        String personInfo = DsHttpClient.getRequest(person_url + token);
        log.info("调用获取个人身份信息接口返回数据：{}", personInfo);
        if (StringUtils.isBlank(personInfo) || JSONObject.parseObject(personInfo).size()
== 0 || JSONObject.parseObject(personInfo).containsKey("error")) {
```

```
            throw new GlobalException("获取个人身份信息异常");
        }
        JSONObject personContent = JSONObject.parseObject(personInfo);
        String account = personContent.getString("account");
        SysUserEntity user = this.getUserByName(account, login.getSystemCode());
        // 不存在则增加用户
        if (user == null) {
            user = new SysUserEntity(IdUtil.getSnowflake().nextId(), account, personConte
nt.getString("name"), DigestUtil.md5Hex(account.substring(0, 1).toUpperCase() + account.s
ubstring(1) + "12#$"), personContent.getString("mobile")
                    , personContent.getString("email"), 1L);
            List<Long> roleId = sysRoleService.list(new LambdaQueryWrapper<SysRoleEntity>
().eq(SysRoleEntity::getSystemCode, login.getSystemCode())
                    .eq(SysRoleEntity::getRoleCode, SystemConstant.REGULAR_USER).orderByA
sc(SysRoleEntity::getId)).stream().map(SysRoleEntity::getId).collect(Collectors.toList());
            // 保存用户
            this.saveUser(user, roleId.get(0).toString(), login.getSystemCode());
        }
        if (StringUtils.isBlank(user.getRoleCode())) {
            // 角色默认普通用户
            user.setRoleCode(SystemConstant.REGULAR_USER);
        }
        Map<String, Object> map = new HashMap<>();
        // 将 token 放入 redis 中
        redisTemplate.opsForValue().set(SystemConstant.USER_TOKEN_REDIS_KEY + token, user.g
etId() + "#" + login.getSystemCode() + "#" + user.getRoleCode(), time, TimeUnit.SECONDS);
        map.put("access_token", token);
        map.put("expire", time);
        map.put("systemCode", login.getSystemCode());
        map.put("userInfo", new SysUserEntity(user.getId(), user.getUsername(), user.getR
ealname(), user.getDepartmentId(), user.getSysDefault(), user.getRoleCode()));
        return map;
    }
}
package com.cy.ftamlirtp.modules.admin.vo.login;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Data;
import javax.validation.constraints.NotEmpty;
@Data
@ApiModel(description = "统一登录对象")
public class LoginBigDataReqVo {

    @ApiModelProperty(value = "授权码")
    @NotEmpty(message = "授权码不能为空")
    private String code;

    @ApiModelProperty(value = "客户端应用回调地址")
    @NotEmpty(message = "客户端应用回调地址不能为空")
    private String redirectUrl;
    @ApiModelProperty(value = "子系统系统编码")
    @NotEmpty(message = "子系统系统编码不能为空")
    private String systemCode;

}
```