前三十页

```
packagecom.g4b.pulin.service.impl;
importcn.afterturn.easypoi.excel.entity.ExportParams;
importcn.hutool.core.bean.BeanUtil;
importcn.hutool.core.util.ObjectUtil;
importcom.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;
importcom.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
importcom.baomidou.mybatisplus.extension.plugins.pagination.Page;
importcom.baomidou.mybatisplus.extension.service.impl.ServiceImpl;
importcom.g4b.pulin.entity.PlAnalysisCompany;
importcom.g4b.pulin.entity.PlAnalysisProject;
importcom.g4b.pulin.entity.PlCompanyProject;
importcom.g4b.pulin.entity.dto.SubcontractorDto;
importcom.g4b.pulin.enums.PlAnalysisCompanyExceptionEnum;
importcom.g4b.pulin.fastdfs.FileUtil;
importcom.g4b.pulin.fastdfs.consts.FileConstantEnum;
importcom.g4b.pulin.fastdfs.dto.FileDTO;
importcom.g4b.pulin.mapper.PlAnalysisCompanyMapper;
importcom.g4b.pulin.param.PlAnalysisCompanyParam;
importcom.g4b.pulin.service.PlAnalysisCompanyService;
importcom.g4b.pulin.service.PlAnalysisProjectService;
importcom.g4b.pulin.service.PlCompanyProjectService;
importlombok.extern.slf4j.Slf4j;
importorg.apache.poi.hssf.usermodel.*;
importorg.apache.poi.ss.usermodel.BorderStyle;
importorg.apache.poi.ss.usermodel.HorizontalAlignment;
importorg.apache.poi.ss.usermodel.VerticalAlignment;
importorg.apache.poi.ss.util.CellRangeAddress;
importorg.springframework.beans.BeanUtils;
importorg.springframework.beans.factory.annotation.Autowired;
importorg.springframework.stereotype.Service;
importorg.springframework.transaction.annotation.Transactional;
importvip.xiaonuo.core.exception.ServiceException;
importvip.xiaonuo.core.factory.PageFactory;
importvip.xiaonuo.core.pojo.page.PageResult;
importvip.xiaonuo.core.util.PageUtil;
importjavax.annotation.Resource;
importjava.io.ByteArrayInputStream;
importjava.io.ByteArrayOutputStream;
importjava.io.IOException;
importjava.text.DecimalFormat;
importjava.util.ArrayList;
importjava.util.List;
```

```
/**
*<p>
*检测公司表服务实现类
*</p>
*
*@authorg4b.pulin
*@since2023-09-18
*/
@Slf4j
@Service
publicclassPlAnalysisCompanyServiceImplextendsServiceImpl<PlAnalysisCompanyMapper,PlAnalysis
Company>implementsPlAnalysisCompanyService{
@Autowired
privatePlAnalysisProjectServiceplAnalysisProjectService;
@Autowired
privatePlCompanyProjectServiceplCompanyProjectService;
@Resource
privatePlAnalysisCompanyMapperplAnalysisCompanyMapper;
@Override
publicPlAnalysisCompanygetByName(Stringname){
LambdaQueryWrapper<PlAnalysisCompany>plAnalysisCompanyLambdaQueryWrapper=newLambdaQueryWrapp
er<>();
plAnalysisCompanyLambdaQueryWrapper.eq(PlAnalysisCompany::getCompanyName,name);
plAnalysisCompanyLambdaQueryWrapper.eq(PlAnalysisCompany::getIsDeleted,0);
PlAnalysisCompanyplAnalysisCompany=this.getOne(plAnalysisCompanyLambdaQueryWrapper);
returnplAnalysisCompany;
}
//分包单位
@Override
publicPageResult<SubcontractorDto>subcontractorPage(PlAnalysisCompanyParamplAnalysisCompanyP
aram){
LambdaQueryWrapper<PlAnalysisCompany>plAnalysisCompanyLambdaQueryWrapper=newLambdaQueryWrapp
er<>();
if(plAnalysisCompanyParam!=null){
if(plAnalysisCompanyParam.getCompanyName()!=null&&plAnalysisCompanyParam.getCompanyName()!="
"){
plAnalysisCompanyLambdaQueryWrapper.like(PlAnalysisCompany::getCompanyName,plAnalysisCompany
Param.getCompanyName());
}
if(plAnalysisCompanyParam.getId()!=null&&plAnalysisCompanyParam.getId()!=""){
plAnalysisCompanyLambdaQueryWrapper.eq(PlAnalysisCompany::getId,plAnalysisCompanyParam.getId
());
}
}
```

```
plAnalysisCompanyLambdaQueryWrapper.eq(PlAnalysisCompany::getIsDeleted,0);
plAnalysisCompanyLambdaQueryWrapper.orderByDesc(PlAnalysisCompany::getCreateTime);
List<PlAnalysisCompany>plAnalysisCompanies=this.list(plAnalysisCompanyLambdaQueryWrapper);
//封装 dto
List<SubcontractorDto>subcontractorDtos=newArrayList<>();
for(PlAnalysisCompanyplAnalysisCompany:plAnalysisCompanies){
SubcontractorDtosubcontractorDto=newSubcontractorDto();
BeanUtils.copyProperties(plAnalysisCompany,subcontractorDto);
//由于该接口接收的是 parm 的对象
PlAnalysisCompanyParamplAnalysisCompanyParam1=newPlAnalysisCompanyParam();
plAnalysisCompanyParam1.setId(plAnalysisCompany.getId());
//获取到公司对应的检查项目
List<PlAnalysisProject>plAnalysisProject=plAnalysisProjectService.getByAnalysisCompany(plAna
lysisCompanyParam1);
subcontractorDto.setAnalysisProject(plAnalysisProject);
subcontractorDtos.add(subcontractorDto);
}
Page<SubcontractorDto>page1=PageFactory.defaultPage();
page1.setTotal(subcontractorDtos.size());
List<SubcontractorDto>resultList=PageUtil.page(page1,subcontractorDtos);
returnnewPageResult(page1,resultList);
}
@Override
publicvoidsaveSubcontractor(SubcontractorDtosubcontractorDto){
PlAnalysisCompanyplAnalysisCompany=newPlAnalysisCompany();
BeanUtils.copyProperties(subcontractorDto,plAnalysisCompany);
PlAnalysisCompanyplAnalysisCompanyByName=getByName(subcontractorDto.getCompanyName());
//有重名的分包单位(全称一样电话一样负责人一样)
if(plAnalysisCompanyByName!=null
&&plAnalysisCompanyByName.getCompanyName().equals(subcontractorDto.getCompanyName())
&&plAnalysisCompanyByName.getPhone().equals(subcontractorDto.getPhone())
&&plAnalysisCompanyByName.getDirector().equals(subcontractorDto.getDirector())
){
//进行检测项目累加
//新增检测项目
List<PlAnalysisProject>analysisProjects=subcontractorDto.getAnalysisProject();
plAnalysisProjectService.addsAnalysisProjectByCompany(analysisProjects,plAnalysisCompanyByNa
me.getId());
}else{
//封装检测公司下的检测项目
//新增检查公司
this.save(plAnalysisCompany);//调用 save 方法会自动填充 id
//新增检测项目
List<PlAnalysisProject>analysisProjects=subcontractorDto.getAnalysisProject();
```

```
plAnalysisProjectService.addsAnalysisCompany(analysisProjects,plAnalysisCompany.getId());
}
}
@Override
publicbooleanhasAnalysisProjectById(Stringid,StringprojectName){
Integercount=plAnalysisCompanyMapper.countAnalysisProjectById(id,projectName);
returncount>0;
}
@Override
publicPageResult<PlAnalysisCompany>page(PlAnalysisCompanyParamplAnalysisCompanyParam){
QueryWrapper<PlAnalysisCompany>queryWrapper=newQueryWrapper<>();
if(ObjectUtil.isNotNull(plAnalysisCompanyParam)){
//根据检测公司简称查询
if(ObjectUtil.isNotEmpty(plAnalysisCompanyParam.getCompanyAbbreviation())){
queryWrapper.lambda().eq(PlAnalysisCompany::getCompanyAbbreviation,plAnalysisCompanyParam.ge
tCompanyAbbreviation());
}
//根据检测公司完整名称查询
if(ObjectUtil.isNotEmpty(plAnalysisCompanyParam.getCompanyName())){
queryWrapper.lambda().eq(PlAnalysisCompany::getCompanyName,plAnalysisCompanyParam.getCompany
Name());
}
//根据负责人查询
if(ObjectUtil.isNotEmpty(plAnalysisCompanyParam.getDirector())){
queryWrapper.lambda().eq(PlAnalysisCompany::getDirector,plAnalysisCompanyParam.getDirector()
);
}
//根据联系方式查询
if(ObjectUtil.isNotEmpty(plAnalysisCompanyParam.getPhone())){
queryWrapper.lambda().eq(PlAnalysisCompany::getPhone,plAnalysisCompanyParam.getPhone());
}
//根据是否删除，0为未删除，1为已删除查询
if(ObjectUtil.isNotEmpty(plAnalysisCompanyParam.getIsDeleted())){
queryWrapper.lambda().eq(PlAnalysisCompany::getIsDeleted,plAnalysisCompanyParam.getIsDeleted
());
}
}
returnnewPageResult<>(this.page(PageFactory.defaultPage(),queryWrapper));
}
@Override
publicList<PlAnalysisCompany>list(PlAnalysisCompanyParamplAnalysisCompanyParam){
returnthis.list();
}
@Override
```

```
publicvoidadd(PlAnalysisCompanyParamplAnalysisCompanyParam){
PlAnalysisCompanyplAnalysisCompany=newPlAnalysisCompany();
BeanUtil.copyProperties(plAnalysisCompanyParam,plAnalysisCompany);
this.save(plAnalysisCompany);
}
@Transactional(rollbackFor=Exception.class)
@Override
publicvoiddelete(PlAnalysisCompanyParamplAnalysisCompanyParam){
Integercount=plAnalysisCompanyMapper.check(plAnalysisCompanyParam.getId());
if(count>0){
thrownewServiceException(PlAnalysisCompanyExceptionEnum.CANNOT_DELETE);
}
PlAnalysisCompanyplAnalysisCompany=newPlAnalysisCompany();
BeanUtils.copyProperties(plAnalysisCompanyParam,plAnalysisCompany);
this.removeById(plAnalysisCompany);
//获取到旗下的检测项目
LambdaQueryWrapper<PlCompanyProject>plCompanyProjectLambdaQueryWrapper=newLambdaQueryWrapper
<>();
plCompanyProjectLambdaQueryWrapper.eq(PlCompanyProject::getCompanyId,plAnalysisCompany.getId
());
List<PlCompanyProject>projects=plCompanyProjectService.list(plCompanyProjectLambdaQueryWrapp
er);
//删除关联表
plCompanyProjectService.remove(plCompanyProjectLambdaQueryWrapper);
//删除检测项目
for(PlCompanyProjectplCompanyProject:projects){
plAnalysisProjectService.removeById(plCompanyProject);
}
}
@Transactional(rollbackFor=Exception.class)
@Override
publicvoidedit(PlAnalysisCompanyParamplAnalysisCompanyParam){
PlAnalysisCompanyplAnalysisCompany=this.queryPlAnalysisCompany(plAnalysisCompanyParam);
BeanUtil.copyProperties(plAnalysisCompanyParam,plAnalysisCompany);
this.updateById(plAnalysisCompany);
}
@Override
publicPlAnalysisCompanydetail(PlAnalysisCompanyParamplAnalysisCompanyParam){
returnthis.queryPlAnalysisCompany(plAnalysisCompanyParam);
}
/**
*获取检测公司表
*
*@authorcaizepei
```

```
*@date2023-09-1416:50:37
*/
privatePlAnalysisCompanyqueryPlAnalysisCompany(PlAnalysisCompanyParamplAnalysisCompanyParam)
{
PlAnalysisCompanyplAnalysisCompany=this.getById(plAnalysisCompanyParam.getId());
if(ObjectUtil.isNull(plAnalysisCompany)){
thrownewServiceException(PlAnalysisCompanyExceptionEnum.NOT_EXIST);
}
returnplAnalysisCompany;
}
@Override
publicList<String>export(PlAnalysisCompanyParamplAnalysisCompanyParam){
List<SubcontractorDto>subcontractorDtoList=newArrayList<>();
//有选中的
if(plAnalysisCompanyParam.getIds()==null){
PlAnalysisCompanyParamplAnalysisCompanyParam1=newPlAnalysisCompanyParam();
PageResult<SubcontractorDto>subcontractorDtoPageResult=this.subcontractorPage(plAnalysisComp
anyParam1);
subcontractorDtoList=subcontractorDtoPageResult.getRows();
}else{
for(Stringid:plAnalysisCompanyParam.getIds()){
plAnalysisCompanyParam.setId(id);
PageResult<SubcontractorDto>subcontractorDtoPageResult=this.subcontractorPage(plAnalysisComp
anyParam);
List<SubcontractorDto>SubcontractorDtolist=subcontractorDtoPageResult.getRows();
SubcontractorDtosubcontractorDto=SubcontractorDtolist.get(0);
subcontractorDtoList.add(subcontractorDto);
}
}
//String[]keys={"分包单位","分包单位名称","负责人","联系方式","检测项目","创建时间"};
String[]keys={"编号","检测项目","检测金额","检测方法","检测指标","检测周期","样品数量"};
//改为一个公司导出一个 excel
ArrayList<String>urls=newArrayList<>();
for(SubcontractorDtosubcontractorDto:subcontractorDtoList){
List<SubcontractorDto>subcontractorDtoList2=newArrayList<>();
subcontractorDtoList2.add(subcontractorDto);
Stringurl=excelExport(newExportParams(),subcontractorDtoList2,keys);
urls.add(url);
}
returnurls;
}
publicStringexcelExport(ExportParamsentity,
List<SubcontractorDto>SubcontractorDtolist,String[]keys){
//创建一个工作簿
```

```
HSSFWorkbookworkbook=newHSSFWorkbook();
//创建一个 sheet
HSSFSheetsh=workbook.createSheet("sheetName");
//列宽
sh.setColumnWidth(0,3766);
sh.setDefaultRowHeight((short)500);
//创建单元格样式(全局样式)
HSSFCellStylecellStyle=workbook.createCellStyle();
//设置单元格水平居中
cellStyle.setAlignment(HorizontalAlignment.CENTER);
//设置单元格垂直居中
cellStyle.setVerticalAlignment(VerticalAlignment.CENTER);
cellStyle.setBorderBottom(BorderStyle.THIN);//下边框
cellStyle.setBorderLeft(BorderStyle.THIN);//左边框
cellStyle.setBorderTop(BorderStyle.THIN);//上边框
cellStyle.setBorderRight(BorderStyle.THIN);//右边框
HSSFFontfont2=workbook.createFont();
font2.setFontHeightInPoints((short)12);
cellStyle.setFont(font2);
//标题样式
HSSFCellStylecellStyleTitle=workbook.createCellStyle();
cellStyleTitle.setAlignment(HorizontalAlignment.CENTER);
cellStyleTitle.setVerticalAlignment(VerticalAlignment.CENTER);
cellStyleTitle.setBorderBottom(BorderStyle.THIN);//下边框
cellStyleTitle.setBorderLeft(BorderStyle.THIN);//左边框
cellStyleTitle.setBorderTop(BorderStyle.THIN);//上边框
cellStyleTitle.setBorderRight(BorderStyle.THIN);//右边框
//标题字体
HSSFFontfont=workbook.createFont();
font.setBold(true);
font.setFontHeightInPoints((short)14);
cellStyleTitle.setFont(font);
//创建 Excel 工作表第一行,设置表头信息
HSSFRowrow0=sh.createRow(0);
introwIndex=0;//行索引(当前行)
for(SubcontractorDtosubcontractorDto:SubcontractorDtolist){
//获取检测项目数量
intprojectSize=subcontractorDto.getAnalysisProject().size();
//检测公司
HSSFRowcompanyRow=sh.createRow(++rowIndex);
for(inti=0;i<keys.length;i++){
HSSFCellcompanyCell=companyRow.createCell(i);
if(i==0){
companyCell.setCellValue("检测公司名称");
```

```
}elseif(i==3){
companyCell.setCellValue(subcontractorDto.getCompanyAbbreviation());
}
companyRow.getCell(i).setCellStyle(cellStyleTitle);
}
//合并列
sh.addMergedRegion(newCellRangeAddress(rowIndex,rowIndex,0,2));
sh.addMergedRegion(newCellRangeAddress(rowIndex,rowIndex,3,6));
//标题
HSSFRowtitleRow=sh.createRow(++rowIndex);
for(inti=0;i<keys.length;i++){
//设置单元格宽度
sh.setColumnWidth(i,256*50+184);
HSSFCellcell=titleRow.createCell(i);
cell.setCellValue(keys[i]);
cell.setCellStyle(cellStyleTitle);
}
//记录检测项目名称
StringprojectName=null;
//序号
intk=0;
//每个检测项目单独创建一行
for(inti=0;i<projectSize;i++){//行
//记录检测项目名称
HSSFRowrow=sh.createRow(++rowIndex);
//给这行的每列写入数据
for(intj=0;j<keys.length;j++){//列
//设置单元格宽度
sh.setColumnWidth(j,256*30);
HSSFCellcell=row.createCell(j);
Stringvalue="";
switch(j){
//TODO导出按项目名称合并行未实现
case0:
//if(projectName!=null&&projectName.equals(String.valueOf(subcontractorDto.getAnalysisProjec
t().get(i).getProjectName()))){
//value=String.valueOf(k);
//break;
//}else{
value=String.valueOf(++k);//编号
break;
//}
case1:
value=String.valueOf(subcontractorDto.getAnalysisProject().get(i).getProjectName());//检测项
```

```
目
////如果检测项目名称相同合并(比如我要合并第二行到第四行的第六列到第八列)
//if(projectName!=null&&projectName.equals(String.valueOf(subcontractorDto.getAnalysisProject().get(i).getProjectName()))){
//log.info("序号：第"+(rowIndex-1)+"与"+rowIndex+"行");
//sh.addMergedRegion(newCellRangeAddress(rowIndex-1,rowIndex,0,0));//合并序号
//log.info("项目名称：第"+(rowIndex-1)+"与"+rowIndex+"行");
//sh.addMergedRegion(newCellRangeAddress(rowIndex-1,rowIndex,1,1));//合并项目名称
//}
projectName=value;
break;
case2:
DecimalFormatamountInYuan=newDecimalFormat("#.00");
Stringformat=amountInYuan.format(subcontractorDto.getAnalysisProject().get(i).getAmount()/100.00);
value=String.valueOf(format+"元");//检测金额
break;
case3:
value=String.valueOf(subcontractorDto.getAnalysisProject().get(i).getTestMethod());//检测方法
break;
case4:
value=String.valueOf(subcontractorDto.getAnalysisProject().get(i).getAnalysisIndex());//检测指标
break;
case5:
value=String.valueOf(subcontractorDto.getAnalysisProject().get(i).getTestPeriod());//检测周期
break;
case6:
if(subcontractorDto.getAnalysisProject().get(i).getSampleSum()==null){
value="";
}else{
value=String.valueOf(subcontractorDto.getAnalysisProject().get(i).getSampleSum());//检测数量
}
break;
default:
break;
}
cell.setCellValue(value);
cell.setCellStyle(cellStyle);
}
}
}
```

```
ByteArrayInputStreamin=null;
ByteArrayOutputStreamos=null;
FileDTOfileDTO=null;
try{
os=newByteArrayOutputStream();
workbook.write(os);
byte[]b=os.toByteArray();
in=newByteArrayInputStream(b);
os.close();
fileDTO=FileUtil.uploadFile(in,false,FileConstantEnum.TEMP.getScene(),"SnowyPlSample.xlsx");
}catch(Exceptione){
e.printStackTrace();
}finally{
if(in!=null){
try{
in.close();
}catch(IOExceptione){
e.printStackTrace();
}
}
if(os!=null){
try{
os.close();
}catch(IOExceptione){
e.printStackTrace();
}
}
}
returnfileDTO.getUrl();
}
}
@Transactional(rollbackFor=Exception.class)
@Override
publicvoiddelete(PlAnalysisProjectParamplAnalysisProjectParam){
List<PlAnalysisProject>analysisProjects=plAnalysisProjectParam.getAnalysisProject();
analysisProjects.forEach(analysisProject->{
//（软删该检测项目与检测公司的关联表）
LambdaQueryWrapper<PlCompanyProject>plCompanyProjectLambdaQueryWrapper=newLambdaQueryWrapper
<>();
plCompanyProjectLambdaQueryWrapper.eq(PlCompanyProject::getProjectId,analysisProject.getId()
);
plCompanyProjectService.remove(plCompanyProjectLambdaQueryWrapper);
this.removeById(analysisProject.getId());
});
```

```
}
@Transactional(rollbackFor=Exception.class)
@Override
publicvoidedit(PlAnalysisProjectParamplAnalysisProjectParam){
PlAnalysisProjectplAnalysisProject=this.queryPlAnalysisProject(plAnalysisProjectParam);
BeanUtil.copyProperties(plAnalysisProjectParam,plAnalysisProject);
this.updateById(plAnalysisProject);
}
@Override
publicPlAnalysisProjectdetail(PlAnalysisProjectParamplAnalysisProjectParam){
returnthis.queryPlAnalysisProject(plAnalysisProjectParam);
}
/**
*获取检测项目表
*
*@authorcaizepei
*@date2023-09-1416:51:58
*/
privatePlAnalysisProjectqueryPlAnalysisProject(PlAnalysisProjectParamplAnalysisProjectParam)
{
PlAnalysisProjectplAnalysisProject=this.getById(plAnalysisProjectParam.getId());
if(ObjectUtil.isNull(plAnalysisProject)){
thrownewServiceException(PlAnalysisProjectExceptionEnum.NOT_EXIST);
}
returnplAnalysisProject;
}
/**
*导入检测项目表_不入库只解释返回
*
*@authorcaizepei
*@date2023-09-1416:51:58
*/
@Override
publicList<PlAnalysisProject>importExcel(MultipartFilefile)throwsIOException{
//if(file.isEmpty()){
////TODO抛出准确异常
//thrownewServiceException(PlAnalysisProjectExceptionEnum.NOT_EXIST);
//}
//List<PlAnalysisProjectExcel>plAnalysisProjectExcels=PoiUtil.importExcel(file,0,1,PlAnalysi
sProjectExcel.class);
//if(plAnalysisProjectExcels==null){
////TODO抛出准确异常
//thrownewServiceException(PlAnalysisProjectExceptionEnum.DATA_EMPTG);
//}
```

```
//List<PlAnalysisProject>plAnalysisProjectList=newArrayList<>();
//for(PlAnalysisProjectExcelitem:plAnalysisProjectExcels){
//PlAnalysisProjectplAnalysisProject=newPlAnalysisProject();
//BeanUtils.copyProperties(item,plAnalysisProject);
////价格转为分给前端
//BigDecimalbigDecimal=newBigDecimal(item.getAmount());
//BigDecimalmultiply=bigDecimal.multiply(BigDecimal.valueOf(100));
//plAnalysisProject.setAmount(Integer.parseInt(multiply.toString()));
//plAnalysisProjectList.add(plAnalysisProject);
//}
//returnplAnalysisProjectList;
//要返回的列表
List<PlAnalysisProject>plAnalysisProjectList=newArrayList<>();
//获取到 workbook 对象
Workbookworkbook=PoiExcelUtils.creatWorkBook(file);
//循环获取到多个 sheet 表
for(intsheetNum=0;sheetNum<workbook.getNumberOfSheets();sheetNum++){
Sheetsheet=workbook.getSheetAt(sheetNum);//获取表格
//校验 sheet 是否合法
if(sheet==null){
continue;
}
RowfirstRow=sheet.getRow(sheet.getFirstRowNum());//获取第一行，一般是标题
introwStart=1;//第三行开始遍历,数据起始行
introwEnd=sheet.getPhysicalNumberOfRows();//获取有记录的行数，即：最后有数据的行是第 n 行，前面有 m 行是空行没数据，则返回 n-m
//记录当前行的项目名称
StringprojectName=null;

for(introwNum=rowStart;rowNum<rowEnd;rowNum++){//行
Rowrow=sheet.getRow(rowNum);
log.info("第{}行",row.getRowNum());
if(null==row){
continue;
}
//检测项目
PlAnalysisProjectplAnalysisProject=newPlAnalysisProject();
for(intcellNum=0;cellNum<7;cellNum++){//列
Cellcell=row.getCell(cellNum);
if(cell!=null){
Stringcontent=PoiExcelUtils.convertCellValueToString(cell);
log.info("第"+row.getRowNum()+"行，第"+cellNum+"列，内容："+content);
//TODO 处理数据
switch(cellNum){
```

```
case1://检测项目名称
plAnalysisProject.setProjectName(content);
//如果项目名称为空则代表合并了
if(content==null){
plAnalysisProject.setProjectName(projectName);
}elseif(content!=null){
projectName=content;
}
break;
case2://检测金额(元转分)
try{
BigDecimalbigDecimal=newBigDecimal(content).setScale(2);
intamount=bigDecimal.multiply(newBigDecimal(100)).intValue();
plAnalysisProject.setAmount(amount);
}catch(NumberFormatExceptione){
//处理异常，例如打印错误消息或返回默认值
thrownewServiceException(4000,"检测金额不需要添加单位，默认单位为元:");
}
break;
case3://检测方法
plAnalysisProject.setTestMethod(content);
break;
case4://检测指标
plAnalysisProject.setAnalysisIndex(content);
case5://检测周期
plAnalysisProject.setTestPeriod(content);
break;
case6://样品数量
plAnalysisProject.setSampleSum(content);
break;
default:
break;
}
}
}
plAnalysisProjectList.add(plAnalysisProject);
}
}
returnplAnalysisProjectList;
}
@Override
publicvoidexport(HttpServletResponseres,PlAnalysisProjectParamplAnalysisProjectParam)throwsI
OException{
//List<PlAnalysisProject>list=this.list(plAnalysisProjectParam);
```

```
//Stringpath="E:\\yyn\\pulin\\pulin(3)\\pulin\\pulin\\pulin\\src\\main\\resources\\template\
\analysis_project_template.xlsx";
//res.setCharacterEncoding("UTF-8");
////attachment 是以附件的形式下载，inline 是浏览器打开
//res.setContentType("application/vnd.openxmlformats-officedocument.spreadsheetml.sheet;char
set=utf-8");
//res.setHeader("Content-Disposition","attachment;filename=plAnalysisProject_template.xlsx")
;
//res.setHeader("content-type","application/octet-stream");
////res.setHeader("Content-Disposition","attachment;filename=\""+fileName+"\"");
//res.setContentType("application/octet-stream");
//res.setHeader("Access-Control-Allow-Origin","*");
////res.setHeader("Content-Disposition","attachment;filename="+URLEncoder.encode(excelName))
;
//ServletOutputStreamos=res.getOutputStream();
//Filefile=newFile(path);
//byte[]bytes=FileUtils.readFileToByteArray(file);
//os.write(bytes);
//os.flush();
//os.close();
////PoiUtil.exportExcelWithStream("SnowyPlAnalysisProject.xls",PlAnalysisProject.class,list)
;
}
@Override
publicList<PlAnalysisProject>getByAnalysisCompany(PlAnalysisCompanyParamplAnalysisCompanyPar
am){
List<PlAnalysisProject>plAnalysisProjects=plAnalysisProjectMapper.getByAnalysisCompany(plAna
lysisCompanyParam.getId());
returnplAnalysisProjects;
}

@Override
publicvoidaddsAnalysisCompany(List<PlAnalysisProject>analysisProjects,StringPlAnalysisCompan
yId){
//判断是否需要删除检测项目
LambdaQueryWrapper<PlCompanyProject>plCompanyProjectLambdaQueryWrapper=newLambdaQueryWrapper
<>();
plCompanyProjectLambdaQueryWrapper.eq(PlCompanyProject::getCompanyId,PlAnalysisCompanyId);
//获取到检测公司下的检测项目 id
List<PlCompanyProject>plCompanyProjects=plCompanyProjectService.list(plCompanyProjectLambdaQ
ueryWrapper);
//获取到该检测单位下的所有检测项目 id
ArrayList<String>plProjectIds=newArrayList<>();
for(PlCompanyProjectplCompanyProject:plCompanyProjects){
```

```
plProjectIds.add(plCompanyProject.getProjectId());
}
//如果检测项目对象为空则把该检测单位下的检查项目全部清空
if(analysisProjects.size()==0){
//关联表
plCompanyProjectLambdaQueryWrapper.eq(PlCompanyProject::getCompanyId,PlAnalysisCompanyId);
plCompanyProjectService.remove(plCompanyProjectLambdaQueryWrapper);
//检测项目表
if(plProjectIds.size()>0){
LambdaQueryWrapper<PlAnalysisProject>analysisProjectLambdaQueryWrapper=newLambdaQueryWrapper
<>();
analysisProjectLambdaQueryWrapper.in(PlAnalysisProject::getId,plProjectIds);
this.remove(analysisProjectLambdaQueryWrapper);
}
}
//不用删的检查项目 id
ArrayList<String>plProjectIds2=newArrayList<>();
for(PlAnalysisProjectplCompanyProject:analysisProjects){
if(plCompanyProject.getId()!=null){
plProjectIds2.add(plCompanyProject.getId());
}
}
if(plProjectIds2.size()>0){
//删除除这些 id 外的检测项目(关联表)
plCompanyProjectLambdaQueryWrapper.notIn(PlCompanyProject::getProjectId,plProjectIds2);
plCompanyProjectLambdaQueryWrapper.in(PlCompanyProject::getProjectId,plProjectIds);
//关联表删除
if(plProjectIds2.size()>0){
plCompanyProjectService.remove(plCompanyProjectLambdaQueryWrapper);
//删除检测项目表
LambdaQueryWrapper<PlAnalysisProject>analysisProjectLambdaQueryWrapper=newLambdaQueryWrapper
<>();
analysisProjectLambdaQueryWrapper.in(PlAnalysisProject::getId,plProjectIds);
analysisProjectLambdaQueryWrapper.notIn(PlAnalysisProject::getId,plProjectIds2);
this.remove(analysisProjectLambdaQueryWrapper);
}
}
@Transactional(rollbackFor=Exception.class)
@Override
@Deprecated
publicvoidapproveOrder(ApproveApplyDTOapproveCancelOrderDTO){
//获取审批表
PlApproveapprove=this.getById(approveCancelOrderDTO.getApproveId());
StringapproveOpinion=approveCancelOrderDTO.getApproveOpinion();
```

```
//审批原因不为空就填入
if(null!=approveOpinion){
approve.setApproveOpinion(approveOpinion);
}
//无论是否通过都需要查询订单，所以提取出来
QueryWrapper<PlOrder>queryWrapper=newQueryWrapper<>();
queryWrapper.lambda().eq(PlOrder::getId,approve.getOrderId());
//1.审核是否通过并且判断是否为申请作废订单
if(approveCancelOrderDTO.getApproveStatus()==2&&approve.getApproveType()==NoticeTypeEnum.ORD
ER_CANCELLATION.getCode()){
//修改审批表的审批状态
approve.setApproveStatus(2);
approve.setUpdateTime(newDate());
this.updateById(approve);
//修改订单的状态为6：已作废
PlOrderorder=plOrderService.getOne(queryWrapper);
//删除关联的样品以及样品相关的信息
List<PlSampleDto>list=plOrderSampleService.getSampleListByOrderId(order.getId());
PlSampleParamplSampleParam=newPlSampleParam();
plSampleParam.setPlSampleDtos(list);
plSampleService.delete(plSampleParam);
order.setOrderStatus(6);
plOrderService.updateById(order);
SysNoticeParamsysNoticeParam=newSysNoticeParam();
sysNoticeParam.setType(2);
sysNoticeParam.setApproveId(approve.getId());
sysNoticeParam.setContent(approveOpinion);
sysNoticeParam.setStatus(1);
plNoticeService.add(sysNoticeParam);
}
//2.未通过
if(approveCancelOrderDTO.getApproveStatus()==3&&approve.getApproveType()==NoticeTypeEnum.ORD
ER_CANCELLATION.getCode()){
//修改审批表的审批状态
approve.setApproveStatus(3);
approve.setUpdateTime(newDate());
this.updateById(approve);
//将订单状态复原
PlOrderorder=plOrderService.getOne(queryWrapper);
order.setOrderStatus(approve.getOrderStatus());
plOrderService.updateById(order);
//三.填写消息通知表Pl_notice
//PlNoticeParamplNoticeParam=newPlNoticeParam();
//plNoticeParam.setNoticeType("2");
```

```
//plNoticeParam.setNoticeTouserId(approve.getPromoter());
//plNoticeParam.setApproveId(approve.getId());
//plNoticeParam.setNoticeContent("申请订单作废审批未通过，请重新申请！");
//plNoticeParam.setIsRead(0);
//plNoticeParam.setIsDeleted(0);
SysNoticeParamsysNoticeParam=newSysNoticeParam();
sysNoticeParam.setType(2);
sysNoticeParam.setApproveId(approve.getId());
sysNoticeParam.setContent(approveOpinion);
sysNoticeParam.setStatus(1);
plNoticeService.add(sysNoticeParam);
}
}
@Override
@Deprecated
publicvoidapproveChangePeriod(ApproveApplyDTOapproveChangePeriodDTO){
//获取审批表
PlApproveapprove=this.getById(approveChangePeriodDTO.getApproveId());
StringapproveOpinion=approveChangePeriodDTO.getApproveOpinion();
//审批原因不为空就填入
if(null!=approveOpinion||approveOpinion.trim().isEmpty()){
approve.setApproveOpinion(approveOpinion);
}
//无论是否通过都需要查询订单，所以提取出来
QueryWrapper<PlOrder>queryWrapper=newQueryWrapper<>();
queryWrapper.lambda().eq(PlOrder::getId,approve.getOrderId());
//1.判断审核是否通过并且是否为变更期次业务
if(approveChangePeriodDTO.getApproveStatus()==2&&approve.getApproveType()==NoticeTypeEnum.CH
ANGE_PERIOD.getCode()){
//修改审批表的审批状态
approve.setApproveStatus(2);
approve.setUpdateTime(newDate());
this.updateById(approve);
//修改订单的状态为1:已录入下单信息
PlOrderorder=plOrderService.getOne(queryWrapper);
order.setOrderStatus(1);
plOrderService.updateById(order);
//三.填写消息通知表Pl_notice
//PlNoticeParamplNoticeParam=newPlNoticeParam();
//plNoticeParam.setNoticeType("3");
//plNoticeParam.setNoticeTouserId(approve.getPromoter());
//plNoticeParam.setApproveId(approve.getId());
//plNoticeParam.setNoticeContent("申请变更期次信息审批已通过");
//plNoticeParam.setIsRead(0);
```

```
//plNoticeParam.setIsDeleted(0);
SysNoticeParamsysNoticeParam=newSysNoticeParam();
sysNoticeParam.setType(3);
sysNoticeParam.setApproveId(approve.getId());
sysNoticeParam.setContent(approveOpinion);
sysNoticeParam.setStatus(1);
plNoticeService.add(sysNoticeParam);
}
//2.未通过
if(approveChangePeriodDTO.getApproveStatus()==3&&approve.getApproveType()==NoticeTypeEnum.CH
ANGE_PERIOD.getCode()){
//修改审批表的审批状态
approve.setApproveStatus(3);
approve.setUpdateTime(newDate());
this.updateById(approve);
//将订单状态复原
PlOrderorder=plOrderService.getOne(queryWrapper);
order.setOrderStatus(approve.getOrderStatus());
plOrderService.updateById(order);
//三.填写消息通知表Pl_notice
//PlNoticeParamplNoticeParam=newPlNoticeParam();
//plNoticeParam.setNoticeType("3");
//plNoticeParam.setNoticeTouserId(approve.getPromoter());
//plNoticeParam.setApproveId(approve.getId());
//plNoticeParam.setNoticeContent("申请变更期次信息审批未通过，请重新申请！");
//plNoticeParam.setIsRead(0);
//plNoticeParam.setIsDeleted(0);
SysNoticeParamsysNoticeParam=newSysNoticeParam();
sysNoticeParam.setType(3);
sysNoticeParam.setApproveId(approve.getId());
sysNoticeParam.setContent(approveOpinion);
sysNoticeParam.setStatus(1);
plNoticeService.add(sysNoticeParam);
}
}
/**
*保存审批表记录
*
*@paramplApprove
*@return
*/
@Override
publicStringsavePlApprove(PlApproveplApprove){
SysLoginUsersysLoginUser=LoginContextHolder.me().getSysLoginUser();
```

```
plApprove.setCreateTime(newDate());
plApprove.setCreateUser(sysLoginUser.getId());
this.save(plApprove);
returnplApprove.getId();
    }
@Override
publicPageResult<PlBillVO>query(BillDTObillDTO){
List<PlBillVO>billVOList=plBillMapper.query(billDTO);
Page<PlBillVO>page=PageFactory.defaultPage();
page.setTotal(billVOList.size());
List<PlBillVO>plBillPage=PageUtil.page(page,billVOList);
for(PlBillVOplBillVO:plBillPage){
List<PlPeriodVO>plPeriodVOList=plBillDetailService.queryPeriodByBid(plBillVO.getId());
plBillVO.setPlPeriodVOList(plPeriodVOList);
}
//该订单的客户是否有关联合同（不需要这功能了）
//plBillPage.forEach(o->o.setHasContract(plCustomerService.customerHasContract(o.getCustomer
Name())));
plBillPage.forEach(o->o.setHasContract(true));
returnnewPageResult<>(page,plBillPage);
}
/**
*收款管理-录入收款信息
*
*@paramaddBillDTO
*/
@Transactional(rollbackFor=Exception.class)
@Override
publicvoidadd(AddBillDTOaddBillDTO){
SysLoginUsersysLoginUser=LoginContextHolder.me().getSysLoginUser();
LambdaUpdateWrapper<PlBill>billWrapper=newLambdaUpdateWrapper<>();
billWrapper.set(PlBill::getCollection,addBillDTO.getCollection());
billWrapper.set(PlBill::getInvoiceCode,addBillDTO.getInvoiceCode());
billWrapper.set(PlBill::getCollectTime,LocalDateTime.now());
billWrapper.set(PlBill::getPaymentUser,String.valueOf(sysLoginUser.getId()));
billWrapper.set(PlBill::getPaymentMethod,addBillDTO.getPaymentMethod());
billWrapper.set(PlBill::getCollectTime,addBillDTO.getCollectTime());
billWrapper.set(PlBill::getPaymentAccount,addBillDTO.getPaymentAccount());
billWrapper.eq(PlBill::getId,addBillDTO.getId());
//一个账单可能包含多个订单
List<String>orderIds=plBillDetailService.queryOrderIdByBid(addBillDTO.getId());
for(StringorderId:orderIds){
PlOrderPersonnelplOrderPersonnel=newPlOrderPersonnel();
plOrderPersonnel.setRoleType(RoleTypeEnum.FINANCE_ROLE.getType());
```

```
plOrderPersonnel.setUserName(sysLoginUser.getName());
plOrderPersonnel.setUserId(sysLoginUser.getId());
plOrderPersonnel.setOrderId(orderId);
plOrderPersonnel.setIsDeleted(0);
plOrderPersonnelService.save(plOrderPersonnel);
//收款后，同步订单状态
PlOrderplOrder=newPlOrder();
plOrder.setId(orderId);
plOrder.setOrderStatus(OrderStatusEnum.PAYED.getCode());
plOrderService.updateById(plOrder);
}
//如果实收>=应收,自动核销,将期次状态改成已支付,账单改成已核销
if(addBillDTO.getCollection()>=addBillDTO.getReceivable()){
billWrapper.set(PlBill::getBalance,addBillDTO.getCollection()-addBillDTO.getReceivable());
billWrapper.set(PlBill::getWriteOff,1);
billWrapper.set(PlBill::getWriteOffTime,LocalDateTime.now());
}
List<String>periodIdList=plPeriodService.queryIdByBillId(addBillDTO.getId());
plPeriodService.updatePayStatusById(periodIdList,PeriodStatusEnum.PAID.getCode());
//当收款时，同步到 plPayment 表
for(StringperiodId:periodIdList){
PlPaymentplPayment=newPlPayment();
plPayment.setPeriodId(periodId);
plPayment.setAmount(addBillDTO.getCollection());
plPayment.setInvoiceCode(addBillDTO.getInvoiceCode());
plPayment.setPaymentMethod(addBillDTO.getPaymentMethod());
plPaymentService.save(plPayment);
}
this.update(billWrapper);
log.info("账单 id:{}-录入收款信息： {}",addBillDTO.getId(),addBillDTO.toString());
}
/**
*核销管理-录入收款信息
*
*@paramwriteOffQueryDTO
*@return
*/
@Override
publicPageResult<WriteOffVO>writeOffPage(WriteOffQueryDTOwriteOffQueryDTO){
List<WriteOffVO>writeOffVOS=plBillMapper.writeOffPage(writeOffQueryDTO);
Page<WriteOffVO>page=PageFactory.defaultPage();
page.setTotal(writeOffVOS.size());
List<WriteOffVO>writeOffVOSPage=PageUtil.page(page,writeOffVOS);
for(WriteOffVOwriteOffVO:writeOffVOSPage){
```

```
List<PlPeriodVO>plPeriodVOList=plBillDetailService.queryPeriodByBid(writeOffVO.getId());
writeOffVO.setPeriodVOList(plPeriodVOList);
}
returnnewPageResult<>(page,writeOffVOS);
}
/**
*核销管理-手动核销
*
*@paramwriteOffEditDTO
*/
@Override
publicvoidwriteOffEdit(WriteOffEditDTOwriteOffEditDTO){
//结余小于 0，不允许核销
//if(writeOffEditDTO.getBalance()<0){
//thrownewServiceException(PlBillExceptionEnum.BALANCE_LT_0);
//}
//检查是否在申请优惠审批中
LambdaQueryWrapper<PlBill>queryWrapper=newLambdaQueryWrapper<>();
queryWrapper.eq(PlBill::getApplyDiscount,1);
queryWrapper.eq(PlBill::getId,writeOffEditDTO.getBillId());
if(this.count(queryWrapper)>0){
thrownewServiceException(PlBillExceptionEnum.APPLY_DISCOUNT);
}
LambdaUpdateWrpper<PlBill>wrapper=newLambdaUpdateWrapper<>();
SysLoginUsersysLoginUser=LoginContextHolder.me().getSysLoginUser();
//查询该账单下的期次是不是全都已收款，如果是，则将账单核销
List<PlPeriodVO>plPeriodVOS=plBillDetailService.queryPeriodByBid(writeOffEditDTO.getBillId()
);
longcount=plPeriodVOS.stream().map(PlPeriodVO::getPayStatus).filter(o->o.equals(PeriodStatus
Enum.PAID.getMessage())).count();
if(count==plPeriodVOS.size()){
wrapper.set(PlBill::getWriteOff,BillStatusEnum.WRITE_OFF.getCode());
}else{
wrapper.set(PlBill::getWriteOff,BillStatusEnum.UN_WRITE_OFF.getCode());
}
wrapper.set(PlBill::getBalance,writeOffEditDTO.getBalance());
wrapper.set(PlBill::getWriteOffTime,LocalDateTime.now());
//设置核销员
wrapper.set(PlBill::getWriteoffUser,String.valueOf(sysLoginUser.getId()));
wrapper.eq(PlBill::getId,writeOffEditDTO.getBillId());
this.update(wrapper);
plPeriodService.writeOffEdit(writeOffEditDTO);
//设置订单状态为已收款
plBillDetailService.queryOrderIdByBid(writeOffEditDTO.getBillId())
```

```
.forEach(o->{
LambdaUpdateWrapper<PlOrder>updateWrapper=newLambdaUpdateWrapper<>();
updateWrapper.set(PlOrder::getOrderStatus,OrderStatusEnum.PAYED.getCode());
updateWrapper.eq(PlOrder::getId,o);
plOrderService.update(updateWrapper);
//PlOrderPersonnelplOrderPersonnel=newPlOrderPersonnel();
//plOrderPersonnel.setRoleType(RoleTypeEnum.WRITEOFF_ROLE.getType());
//plOrderPersonnel.setUserName(sysLoginUser.getName());
//plOrderPersonnel.setUserId(sysLoginUser.getId());
//plOrderPersonnel.setOrderId(o);
//plOrderPersonnel.setIsDeleted(0);
//plOrderPersonnelService.save(plOrderPersonnel);
});//跟进记录记录核销员
//跟进记录记录核销员
//根据期次 id 找到订单 id
for(StringperiodId:writeOffEditDTO.getPeriodIdList()){
PlOrderPeriodorderByPeriodId=plOrderPeriodService.getOrderByPeriodId(periodId);
//同步 pl_order_personnel 表
//获取当前用户信息
PlOrderPersonnelplOrderPersonnel=newPlOrderPersonnel();
plOrderPersonnel.setRoleType(6);
plOrderPersonnel.setUserName(sysLoginUser.getName());
plOrderPersonnel.setUserId(sysLoginUser.getId());
plOrderPersonnel.setOrderId(orderByPeriodId.getOrderId());
plOrderPersonnel.setPeriodId(periodId);
plOrderPersonnel.setIsDeleted(0);
plOrderPersonnelService.save(plOrderPersonnel);
}
}
@Override
publicList<OrderPaymentDTO>getBillByOrderId(StringorderId){
returnbaseMapper.getBillByOrderId(orderId);
}
/**
*核销管理-申请账单优惠
*
*@paramwriteOffEditDTO
*/
@Transactional(rollbackFor=Exception.class)
@Override
publicvoiddiscount(WriteOffEditDTOwriteOffEditDTO){
SysLoginUsersysLoginUser=LoginContextHolder.me().getSysLoginUser();
//正在申请账单优惠,修改状态
LambdaQueryWrapper<PlBill>queryWrapper=newLambdaQueryWrapper<>();
```

```
queryWrapper.eq(PlBill::getApplyDiscount,1);
queryWrapper.eq(PlBill::getId,writeOffEditDTO.getBillId());
if(this.count(queryWrapper)>0){
thrownewServiceException(PlBillExceptionEnum.REPEAT_APPLY);
}
LambdaUpdateWrapper<PlBill>updateWrapper=newLambdaUpdateWrapper<>();
updateWrapper.set(PlBill::getApplyDiscount,1);
updateWrapper.eq(PlBill::getId,writeOffEditDTO.getBillId());
this.update(updateWrapper);
SysNoticeParamsysNoticeParam=newSysNoticeParam();
//申请优惠金额的类型
sysNoticeParam.setType(NoticeTypeEnum.APPLY_DISCOUNT.getCode());
sysNoticeParam.setBillDiscount(writeOffEditDTO.getDiscount());
Stringtitle=NoticeTypeEnum.APPLY_DISCOUNT.getTemplate().replace("${jsy}",sysLoginUser.getNam
e());
sysNoticeParam.setTitle(title);
sysNoticeParam.setStatus(1);
sysNoticeParam.setBillId(writeOffEditDTO.getBillId());
plNoticeService.add(sysNoticeParam);
}
@Override
publicPageResult<PlCompanyProject>page(PlCompanyProjectParamplCompanyProjectParam){
QueryWrapper<PlCompanyProject>queryWrapper=newQueryWrapper<>();
if(ObjectUtil.isNotNull(plCompanyProjectParam)){
//根据检测项目 id 查询
if(ObjectUtil.isNotEmpty(plCompanyProjectParam.getProjectId())){
queryWrapper.lambda().eq(PlCompanyProject::getProjectId,plCompanyProjectParam.getProjectId()
);
}
//根据是否删除，0 为未删除，1 为已删除查询
if(ObjectUtil.isNotEmpty(plCompanyProjectParam.getIsDeleted())){
queryWrapper.lambda().eq(PlCompanyProject::getIsDeleted,plCompanyProjectParam.getIsDeleted()
);
}
}
returnnewPageResult<>(this.page(PageFactory.defaultPage(),queryWrapper));
}
@Override
publicList<PlCompanyProject>list(PlCompanyProjectParamplCompanyProjectParam){
returnthis.list();
}
@Override
publicvoidadd(PlCompanyProjectParamplCompanyProjectParam){
PlCompanyProjectplCompanyProject=newPlCompanyProject();
```

```
BeanUtil.copyProperties(plCompanyProjectParam,plCompanyProject);
this.save(plCompanyProject);
}
@Transactional(rollbackFor=Exception.class)
@Override
publicvoiddelete(List<PlCompanyProjectParam>plCompanyProjectParamList){
plCompanyProjectParamList.forEach(plCompanyProjectParam->{
this.removeById(plCompanyProjectParam.getId());
this.removeById(plCompanyProjectParam.getCompanyId());
});
}
@Transactional(rollbackFor=Exception.class)
@Override
publicvoidedit(PlCompanyProjectParamplCompanyProjectParam){
PlCompanyProjectplCompanyProject=this.queryPlCompanyProject(plCompanyProjectParam);
BeanUtil.copyProperties(plCompanyProjectParam,plCompanyProject);
this.updateById(plCompanyProject);
}
@Override
publicPlCompanyProjectdetail(PlCompanyProjectParamplCompanyProjectParam){
returnthis.queryPlCompanyProject(plCompanyProjectParam);
}
/**
*获取检测公司检测项目关系表
*
*@authorcaizepei
*@date2023-09-1416:56:16
*/
privatePlCompanyProjectqueryPlCompanyProject(PlCompanyProjectParamplCompanyProjectParam){
PlCompanyProjectplCompanyProject=this.getById(plCompanyProjectParam.getId());
//PlCompanyProjectplCompanyProject=this.getById(plCompanyProjectParam.getCompanyId());
if(ObjectUtil.isNull(plCompanyProject)){
thrownewServiceException(PlCompanyProjectExceptionEnum.NOT_EXIST);
}
returnplCompanyProject;
}
@Override
publicvoidexport(PlCompanyProjectParamplCompanyProjectParam){
List<PlCompanyProject>list=this.list(plCompanyProjectParam);
PoiUtil.exportExcelWithStream("SnowyPlCompanyProject.xls",PlCompanyProject.class,list);
    }
@Override
publicList<PlContract>getAll(StringcsId){
List<PlContract>list;
```

```
//只显示未删除的合同
LambdaQueryWrapper<PlContract>plContractLambdaQueryWrapper=newLambdaQueryWrapper<>();
plContractLambdaQueryWrapper.eq(PlContract::getIsDeleted,0);
if(StringUtils.isEmpty(csId)){
list=this.list(plContractLambdaQueryWrapper);
}else{
plContractLambdaQueryWrapper.eq(PlContract::getCustomerId,csId);
list=this.list(plContractLambdaQueryWrapper);
}
returnlist;
}
@Override
publicList<PlContract>getByCustomerId(StringcsId){
LambdaQueryWrapper<PlContract>plContractLambdaQueryWrapper=newLambdaQueryWrapper<>();
plContractLambdaQueryWrapper.eq(PlContract::getCustomerId,csId);
List<PlContract>list=this.list(plContractLambdaQueryWrapper);
returnlist;
}
@Transactional(rollbackFor=Exception.class)
@Override
publicPlContractDtoadd(ContractParamcontractParam){
//添加合同时对该合同的审批初始化
PlContractplContract=newPlContract();
BeanUtils.copyProperties(contractParam,plContract);
DateTimeFormatterformatter=DateTimeFormatter.ofPattern("yyyy-MM-dd");
LocalDateeffectTime=LocalDate.parse(contractParam.getEffectTime(),formatter);
LocalDateterminationTime=LocalDate.parse(contractParam.getTerminationTime(),formatter);
plContract.setEffectTime(effectTime);
plContract.setTerminationTime(terminationTime);
//生成编码
Stringcode=this.createCode();
StringfileId=contractParam.getFileId();
plContract.setContractCode(code);
plContract.setIsDeleted(0);
//前端把 id 传了过来，这里需要置空才能添加
plContract.setId(null);
if(LocalDate.parse(contractParam.getEffectTime()).compareTo(LocalDate.now())<=0&&
LocalDate.parse(contractParam.getTerminationTime()).compareTo(LocalDate.now())>=0){
plContract.setExpired(0);
}else{
plContract.setExpired(1);
}
//获取客户名称生成合同名称(客户名称+当前时间)
//前端传过来的 id 是客户 id
```

```
PlCustomerplCustomer=null;
if(contractParam.getId()!=null){
plContract.setCustomerId(contractParam.getId());
plCustomer=plCustomerService.getById(contractParam.getId());
}
//说明此时还未客户添加，无客户 id（新增客户的同时新增合同）
if(plCustomer==null){
plContract.setContractTitle(PlContractExceptionEnum.NOT_CONTRACT_NAME.getMessage());
this.save(plContract);
}else{
SimpleDateFormatdate=newSimpleDateFormat("yyyyMMdd");
plContract.setContractTitle(plCustomer.getCompanyName()+date.format(newDate()));
this.save(plContract);
}
//获取该合同的 id
LambdaQueryWrapper<PlContract>plContractLambdaQueryWrapper=newLambdaQueryWrapper<>();
plContractLambdaQueryWrapper.eq(PlContract::getContractCode,plContract.getContractCode());
PlContractpl=this.getOne(plContractLambdaQueryWrapper);
//添加对应的审批记录
PlApproveplApprove=newPlApprove();
plApprove.setContractId(pl.getId());
plApprove.setApproveType(3);
plApprove.setApproveStatus(1);
plApproveService.save(plApprove);
//传合同编号和合同附件 id
//Map<String,String>stringStringMap=newHashMap<>();
//stringStringMap.put("contractCode",plContract.getContractCode());
//stringStringMap.put("fileId",fileId);
//stringStringMap.put("contractId",pl.getId().toString());
PlContractDtoplContractDto=newPlContractDto();
BeanUtils.copyProperties(plContract,plContractDto);
plContractDto.setContractId(pl.getId());
returnplContractDto;
}
//生成合同编号
publicStringcreateCode(){
StringredisKey="pl_contract_day_counter";
//如果 key 不存在，重置
stringRedisTemplate.opsForValue().setIfAbsent(redisKey,"1");
Stringvaule=stringRedisTemplate.opsForValue().get(redisKey);
vaule=String.format("%04d",Integer.parseInt(vaule));
SimpleDateFormatdate=newSimpleDateFormat("yyyyMMdd");
Stringcode=date.format(newDate())+"-"+(String)vaule;
stringRedisTemplate.opsForValue().increment(redisKey);
```

```
returncode;
}
@Override
publicPlContractDtoedit(PlContractParamplContractParam){
//如果重新上传了合同，会返回新的附件。前端带着新的附件 id 进行修改合同。（绑定新上传的 fileid）
//删除旧的合同 fileid 文件。todo
//更新合同
PlContractplContract=newPlContract();
BeanUtils.copyProperties(plContractParam,plContract);
//生效时间和实效时间
DateTimeFormatterformatter=DateTimeFormatter.ofPattern("yyyy-MM-dd");
LocalDateeffectTime=LocalDate.parse(plContractParam.getEffectTime(),formatter);
LocalDateterminationTime=LocalDate.parse(plContractParam.getTerminationTime(),formatter);
plContract.setEffectTime(effectTime);
plContract.setTerminationTime(terminationTime);
this.updateById(plContract);
//传合同编号和合同附件 id
PlContractDtoplContractDto=newPlContractDto();
BeanUtils.copyProperties(plContract,plContractDto);
plContractDto.setContractId(plContract.getId());
returnplContractDto;
}
@Override
publicPlContractDtogetContractDtoById(Stringid){
//LambdaQueryWrapper<PlContract>plContractLambdaQueryWrapper=newLambdaQueryWrapper<>();
//plContractLambdaQueryWrapper.eq(PlContract::getFileId,fileId);
//PlContractplContract=this.getOne(plContractLambdaQueryWrapper);
PlContractplContract=this.getById(id);
if(plContract.getIsDeleted()==1){
thrownewServiceException(40000,"该合同已删除");
}
PlContractDtoplContractDto=newPlContractDto();
BeanUtils.copyProperties(plContract,plContractDto);
PlCustomerplCustomer=plCustomerService.getById(plContract.getCustomerId());
//PlApproveplApprove=plApproveService.getByContractId(plContract.getId());
PlApproveplApprove=plApproveService.getByContractId(id);
if(plCustomer!=null){
plContractDto.setCsName(plCustomer.getCompanyName());
}
plContractDto.setState(plApprove.getApproveStatus());
plContractDto.setCustomerId(plContract.getCustomerId());
//封装合同 url
FileInfofileInfo=fileInfoService.getById(plContractDto.getFileId());
plContractDto.setFileUrl(fileInfo.getFiUrl());
```

```
returnplContractDto;
}
@Override
publicList<PlContractDto>getContractDtoByCustomerId(StringcustomerId){
LambdaQueryWrapper<PlContract>plContractLambdaQueryWrapper=newLambdaQueryWrapper<>();
plContractLambdaQueryWrapper.eq(PlContract::getCustomerId,customerId);
List<PlContract>plContracts=this.list(plContractLambdaQueryWrapper);

List<PlContractDto>plContractDtos=plContracts.stream().map(item->{
PlContractDtoplContractDto=newPlContractDto();
BeanUtils.copyProperties(item,plContractDto);
PlCustomerplCustomer=plCustomerService.getById(item.getCustomerId());
PlApproveplApprove=plApproveService.getByContractId(item.getId());
if(plCustomer!=null){
plContractDto.setCsName(plCustomer.getCompanyName());
}
plContractDto.setState(plApprove.getApproveStatus());
plContractDto.setCustomerId(item.getCustomerId());
plContractDto.setContractId(item.getId());
//封装合同 url
FileInfofileInfo=fileInfoService.getById(plContractDto.getFileId());
plContractDto.setFileUrl(fileInfo.getFiUrl());
returnplContractDto;
}).collect(Collectors.toList());
returnplContractDtos;
}
@Override
publicvoiddeleteById(List<PlContract>plContractParams){
plContractParams.forEach(plContractParam->{
PlContractplContract=newPlContract();
BeanUtils.copyProperties(plContractParam,plContract);
//plContract.setIsDeleted(1);
this.removeById(plContract);
});
}
@Override
publicFileDTOupload(MultipartFilefile){
returnFileUtil.uploadFile(file,true,"2",file.getOriginalFilename());
}
@Override
publicPageResult<PlContractDto>plContractPage(StringCsName,IntegerapproveStatus,StringcontractCode){
List<PlContractDto>plContractDtos=plContractMapper.plContractPage(CsName,approveStatus,contractCode);
```

```
Page<PlContractDto>page1=PageFactory.defaultPage();
page1.setTotal(plContractDtos.size());
List<PlContractDto>resultList=PageUtil.page(page1,plContractDtos);
returnnewPageResult(page1,resultList);
}
@Override
publicPlContractgetByFileId(StringfileId){
if(fileId!=null){
LambdaQueryWrapper<PlContract>plContractLambdaQueryWrapper=newLambdaQueryWrapper<>();
plContractLambdaQueryWrapper.eq(PlContract::getFileId,fileId);
PlContractplContract=this.getOne(plContractLambdaQueryWrapper);
returnplContract;
}
returnnull;
}
@Override
publicPlContractgetByCode(Stringcode){
LambdaQueryWrapper<PlContract>plContractLambdaQueryWrapper=newLambdaQueryWrapper<>();
plContractLambdaQueryWrapper.eq(PlContract::getContractCode,code);
PlContractplContract=this.getOne(plContractLambdaQueryWrapper);
returnplContract;
}
/**
*把当前时间在生效时间和过期时间的合同,expire=0,否则 expire=1
*
*@return
*/
@Override
publicvoidcontractExpired(){
//QueryWrapper<PlContract>queryWrapper=newQueryWrapper<>();
//queryWrapper.and(q->q.gt("DATE_FORMAT(effect_time,'%Y-%m-%d')","DATE_FORMAT(NOW(),'%Y-%m-%
d')"))
//.or(q->q.lt("DATE_FORMAT(termination_time,'%Y-%m-%d')","DATE_FORMAT(NOW(),'%Y-%m-%d')"));
//queryWrapper.eq("expired","0");
List<PlContract>listExp=plContractMapper.contractExpired();
//List<PlContract>listExp=this.list(queryWrapper);
listExp=listExp.stream().peek(o->{
o.setExpired(1);
log.info("合同 id:{},设置为过期",o.getId());
}).collect(Collectors.toList());
this.updateBatchById(listExp);
@Override
publicList<PlCustomerRecordDto>getByCsId(StringcsId){
LambdaQueryWrapper<PlCustomerRecord>plCustomerRecordLambdaQueryWrapper=newLambdaQueryWrapper
```

```
<>();
plCustomerRecordLambdaQueryWrapper.eq(PlCustomerRecord::getCustomerId,csId);
//按时间排序
plCustomerRecordLambdaQueryWrapper.orderByDesc(PlCustomerRecord::getCreateTi
me);
//添加记录对应的用户名
List<PlCustomerRecordDto>plCustomerRecordDtos=newArrayList<>();
List<PlCustomerRecord>list=this.list(plCustomerRecordLambdaQueryWrapper);
for(PlCustomerRecordplCustomerRecord:list){
PlCustomerRecordDtoplCustomerRecordDto=newPlCustomerRecordDto();
BeanUtils.copyProperties(plCustomerRecord,plCustomerRecordDto);
SysUsersysUser=sysUserService.getById(plCustomerRecord.getUserId());
if(sysUser!=null){
Stringname=sysUser.getName();
plCustomerRecordDto.setUserName(name);
}
plCustomerRecordDtos.add(plCustomerRecordDto);
}
returnplCustomerRecordDtos;
}
@Override
publicvoidedit(PlCustomerRecordplCustomerRecord){
this.updateById(plCustomerRecord);
}
@Override
publicvoidadd(PlCustomerRecordParamplCustomerRecordParam,inttype){
PlCustomerRecordplCustomerRecord=newPlCustomerRecord();
BeanUtils.copyProperties(plCustomerRecordParam,plCustomerRecord);
SysLoginUsersysLoginUser=LoginContextHolder.me().getSysLoginUser();

if(type==1&&loginContext.isSuperAdmin()){
plCustomerRecord.setContents("管理员"+sysLoginUser.getName()+"新增客户，未设置业务人员");
this.save(plCustomerRecord);
}
//事件 2：管理员 XXX 新建客户，设置业务员 XXX 跟进该客户
elseif(type==2&&loginContext.isSuperAdmin()){
SysUsersysUser=sysUserService.getById(plCustomerRecordParam.getUserId());
plCustomerRecord.setContents("管理员"+sysLoginUser.getName()+"新增客户，设置业务员
"+sysUser.getName()+"跟进该客户");
this.save(plCustomerRecord);
}
//事件 3：管理员 XXX 设置该客户的业务员 XXX
elseif(type==3&&loginContext.isSuperAdmin()){
SysUsersysUser=sysUserService.getById(plCustomerRecordParam.getUserId());
```

```
returnnewPageResult<>(this.page(PageFactory.defaultPage(),queryWrapper));
}
@Override
publicList<PlOrderPeriod>list(PlOrderPeriodParamplOrderPeriodParam){
returnthis.list();
}
@Override
publicvoidadd(PlOrderPeriodParamplOrderPeriodParam){
PlOrderPeriodplOrderPeriod=newPlOrderPeriod();
BeanUtil.copyProperties(plOrderPeriodParam,plOrderPeriod);
this.save(plOrderPeriod);
}
@Transactional(rollbackFor=Exception.class)
@Override
publicvoiddelete(List<PlOrderPeriodParam>plOrderPeriodParamList){
plOrderPeriodParamList.forEach(plOrderPeriodParam->{
this.removeById(plOrderPeriodParam.getId());
this.removeById(plOrderPeriodParam.getOrderId());
this.removeById(plOrderPeriodParam.getPeriodId());
});
}
@Transactional(rollbackFor=Exception.class)
@Override
publicvoidedit(PlOrderPeriodParamplOrderPeriodParam){
PlOrderPeriodplOrderPeriod=this.queryPlOrderPeriod(plOrderPeriodParam);
BeanUtil.copyProperties(plOrderPeriodParam,plOrderPeriod);
this.updateById(plOrderPeriod);
}
@Override
publicPlOrderPerioddetail(PlOrderPeriodParamplOrderPeriodParam){
returnthis.queryPlOrderPeriod(plOrderPeriodParam);
}
/**
*获取订单期次关联表
*
*@authorcaizepei
*@date2023-09-1417:00:03
*/
privatePlOrderPeriodqueryPlOrderPeriod(PlOrderPeriodParamplOrderPeriodParam){
PlOrderPeriodplOrderPeriod=this.getById(plOrderPeriodParam.getId());
if(ObjectUtil.isNull(plOrderPeriod)){
thrownewServiceException(PlOrderPeriodExceptionEnum.NOT_EXIST);
}
returnplOrderPeriod;
```

```
}
@Override
publicvoidexport(PlOrderPeriodParamplOrderPeriodParam){
List<PlOrderPeriod>list=this.list(plOrderPeriodParam);
PoiUtil.exportExcelWithStream("SnowyPlOrderPeriod.xls",PlOrderPeriod.class,list);
}
@Override
publicList<PlPeriod>getPeriodListByOrderId(StringorderId){
returnbaseMapper.getPeriodListByOrderId(orderId);
}
//根据期次 id 查询订单 id
@Override
publicPlOrderPeriodgetOrderByPeriodId(StringperiodId){
LambdaQueryWrapper<PlOrderPeriod>plOrderPeriodLambdaQueryWrapper=newLambdaQueryWrapper<>();
plOrderPeriodLambdaQueryWrapper.eq(PlOrderPeriod::getPeriodId,periodId);
PlOrderPeriodplOrderPeriod=this.getOne(plOrderPeriodLambdaQueryWrapper);
returnplOrderPeriod;
    }
@Override
publicList<String>getOrderIdsByUserId(StringbusinessUserId){
QueryWrapper<PlOrderPersonnel>wrapper=newQueryWrapper<>();
wrapper.eq("user_id",businessUserId);
wrapper.select("order_id");
List<PlOrderPersonnel>list=baseMapper.selectList(wrapper);
returnlist.stream().map(item->String.valueOf(item.getOrderId())).collect(Collectors.toList()
);
    }
@Override
publicvoidsaveBatchOrderSample(List<String>plSampleIds,StringorderId){
List<PlOrderSample>list=newArrayList<>(plSampleIds.size());
for(StringplSampleId:plSampleIds){
PlOrderSampleplOrderSample=newPlOrderSample();
plOrderSample.setOrderId(orderId);
plOrderSample.setSampleId(plSampleId);
plOrderSample.setIsDeleted(0);
list.add(plOrderSample);
}
this.saveBatch(list);
}
@Override
publicList<PlSampleDto>getSampleListByOrderId(StringorderId){
List<PlSampleDto>sampleList=baseMapper.getSampleListByOrderId(orderId);
//设置检测项目
for(PlSampleDtoitem:sampleList){
```

```
List<PlAnalysisProject>projectList=sampleProjectService.getProjectListBySampleId(item.getId(
));
item.setProject(projectList);
}
returnsampleList;
}
@Override
publicList<PlOrderSample>getBySampleId(StringsampleId){
LambdaQueryWrapper<PlOrderSample>plOrderSampleLambdaQueryWrapper=newLambdaQueryWrapper<>();
plOrderSampleLambdaQueryWrapper.eq(PlOrderSample::getSampleId,sampleId);
returnthis.list(plOrderSampleLambdaQueryWrapper);
    }
/*
importcn.afterturn.easypoi.excel.entity.ExportParams;
importcn.hutool.core.bean.BeanUtil;
importcn.hutool.core.util.ObjectUtil;
importcn.hutool.log.Log;
importcom.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;
importcom.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
importcom.baomidou.mybatisplus.core.conditions.update.UpdateWrapper;
importcom.baomidou.mybatisplus.extension.plugins.pagination.Page;
importcom.baomidou.mybatisplus.extension.service.impl.ServiceImpl;
importcom.g4b.pulin.commons.emun.*;
importcom.g4b.pulin.entity.*;
importcom.g4b.pulin.entity.dto.ApplyOperateDTO;
importcom.g4b.pulin.entity.dto.OrderPaymentDTO;
importcom.g4b.pulin.entity.dto.PlSampleDto;
importcom.g4b.pulin.entity.dto.RecordDto;
importcom.g4b.pulin.entity.excel.PlOrderExcel;
importcom.g4b.pulin.entity.excel.PlSampleExcel;
importcom.g4b.pulin.enums.PlOrderExceptionEnum;
importcom.g4b.pulin.fastdfs.FileUtil;
importcom.g4b.pulin.fastdfs.consts.FileConstantEnum;
importcom.g4b.pulin.fastdfs.dto.FileDTO;
importcom.g4b.pulin.mapper.PlOrderMapper;
importcom.g4b.pulin.mapper.PlSampleMapper;
importcom.g4b.pulin.param.PlOrderParam;
importcom.g4b.pulin.param.PlOrderSimpParam;
importcom.g4b.pulin.service.*;
importcom.g4b.pulin.util.PoiExcelUtils;
importcom.g4b.pulin.vo.*;
importorg.apache.poi.hssf.usermodel.*;
importorg.apache.poi.ss.usermodel.*;
importorg.apache.poi.ss.util.CellRangeAddress;
```

```
importorg.springframework.beans.factory.annotation.Autowired;
importorg.springframework.data.redis.core.RedisTemplate;
importorg.springframework.data.redis.core.StringRedisTemplate;
importorg.springframework.security.core.context.SecurityContextHolder;
importorg.springframework.stereotype.Service;
importorg.springframework.transaction.annotation.Transactional;
importorg.springframework.util.StringUtils;
importorg.springframework.web.multipart.MultipartFile;
importvip.xiaonuo.core.context.login.LoginContext;
importvip.xiaonuo.core.context.login.LoginContextHolder;
importvip.xiaonuo.core.exception.ServiceException;
importvip.xiaonuo.core.factory.PageFactory;
importvip.xiaonuo.core.pojo.login.SysLoginUser;
importvip.xiaonuo.core.pojo.page.PageResult;
importvip.xiaonuo.core.util.PageUtil;
importvip.xiaonuo.sys.modular.auth.service.AuthService;
importvip.xiaonuo.sys.modular.notice.param.SysNoticeParam;
importvip.xiaonuo.sys.modular.org.service.SysOrgService;
importvip.xiaonuo.sys.modular.role.entity.SysRole;
importvip.xiaonuo.sys.modular.role.enums.SysRoleExceptionEnum;
importvip.xiaonuo.sys.modular.role.service.SysRoleService;
importvip.xiaonuo.sys.modular.user.entity.SysUser;
importvip.xiaonuo.sys.modular.user.service.SysUserRoleService;
importvip.xiaonuo.sys.modular.user.service.SysUserService;
importjavax.annotation.Resource;
importjava.io.ByteArrayInputStream;
importjava.io.ByteArrayOutputStream;
importjava.io.IOException;
importjava.sql.Timestamp;
importjava.text.ParseException;
importjava.text.SimpleDateFormat;
importjava.time.Instant;
importjava.time.LocalDate;
importjava.time.ZoneId;
importjava.util.ArrayList;
importjava.util.Date;
importjava.util.List;
importjava.util.Objects;
importjava.util.stream.Collectors;
/**
*订单表service接口实现类
*
*@authorcaizepei
*@date2023-09-1416:59:38
```

```
*/
@Service
publicclassPlOrderServiceImplextendsServiceImpl<PlOrderMapper,PlOrder>implementsPlOrderServi
ce{
@Autowired
privatePlOrderPersonnelServiceplOrderPersonnelService;
@Autowired
privatePlOrderMapperplOrderMapper;
@Autowired
privatePlOrderSampleServiceplOrderSampleService;
@Resource
privateRedisTemplate<String,Integer>redisTemplate;
@Resource
privateStringRedisTemplatestringRedisTemplate;
@Resource
privateSysRoleServicesysRoleService;
@Resourc
privateSysUserRoleServicesysUserRoleService;
@Resource
privatePlSampleServiceplSampleService;
@Resource
privatePlCustomerServiceplCustomerService;
@Resource
privatePlPeriodServiceplPeriodService;
@Resource
privatePlOrderPeriodServiceplOrderPeriodService;
@Resource
privatePlPaymentServiceplPaymentService;
@Resource
privateSysOrgServicesysOrgService;
@Resource
privateLoginContextloginContext;
@Resource
privateAuthServiceauthService;
@Resource
privatevip.xiaonuo.core.context.login.LoginContextLoginContext;
@Autowired
privatePlApproveServiceplApproveService;
@Autowired
privatePlNoticeServiceplNoticeService;
@Autowired
privateSysUserServicesysUserService;
@Autowired
privatePlOrderPeriodServiceorderPeriodService;
```

```
@Autowired
privatePlPaymentServicepaymentService;
@Autowired
privatePlSampleServicesampleService;
@Autowired
privatePlPeriodServiceperiodService;
@Autowired
privatePlBillServicebillService;
@Autowired
privatePlSampleMapperplSampleMapper;
@Autowired
privatePlEntrustCompanyCustomerServiceentrustCompanyCustomerService;
@Autowired
privatePlEntrustCompanyServiceplEntrustCompanyService;
@Autowired
privatePlAnalysisProjectServiceplAnalysisProjectService;
@Autowired
privatePlAnalysisCompanyServiceplAnalysisCompanyService;
@Autowired
privatePlSampleProjectServiceplSampleProjectService;
@Autowired
privatePlCompanyProjectServiceplCompanyProjectService;
privatestaticfinalLoglog=Log.get();
@Override
publicPageResult<PlOrderVo>page(PlOrderParamplOrderParam){
QueryWrapper<PlOrderVo>queryWrapper=newQueryWrapper<>();
//TODO:根据角色判断哪些字段可以查询，得确定好角色码
SysLoginUsersysLoginUser=LoginContextHolder.me().getSysLoginUser();
//查出该角用户拥有的角色
List<Long>userRoleIdList=sysUserRoleService.getUserRoleIdList(sysLoginUser.getId());
if(ObjectUtil.isNotNull(plOrderParam)){
//根据产品名称查询
if(ObjectUtil.isNotEmpty(plOrderParam.getProductName())){
queryWrapper.like("por.product_name",plOrderParam.getProductName());
}
//根据客户名字查询
if(ObjectUtil.isNotEmpty(plOrderParam.getCustomerName())){
queryWrapper.eq("pcr.company_name",plOrderParam.getCustomerName());
}
//根据订单状态（录入下单信息，待确认下单金额，待收款，已收款，服务已完成，已作废）查询
if(ObjectUtil.isNotEmpty(plOrderParam.getOrderStatus())){
queryWrapper.eq("por.order_status",plOrderParam.getOrderStatus());
}
//根据备案状态（待备案，备案中，备案完成）查询
```

```
if(ObjectUtil.isNotEmpty(plOrderParam.getRecordStatus())){
queryWrapper.eq("por.record_status",plOrderParam.getRecordStatus());
}
//根据订单完成日期查询
if(ObjectUtil.isNotEmpty(plOrderParam.getFinishDate())){
queryWrapper.apply("date_format(por.finish_time,'%Y-%m-%d')<={0}",plOrderParam.getFinishDate
());
}
//根据订单完成日期（年月）查询
if(ObjectUtil.isNotEmpty(plOrderParam.getFinishYM())){
queryWrapper.like("por.finish_time",plOrderParam.getFinishYM());
}
//根据下单员名字查询
if(ObjectUtil.isNotEmpty(plOrderParam.getOrderRole())){
queryWrapper.eq("popl.user_name",plOrderParam.getOrderRole());
}
//根据业务员名字查询
if(ObjectUtil.isNotEmpty(plOrderParam.getTransactRole())){
queryWrapper.eq("su.`name`",plOrderParam.getTransactRole());
}
////根据财务员名字查询
//if(ObjectUtil.isNotEmpty(plOrderParam.getTransactRole())){
//queryWrapper.eq("pop3.user_name",plOrderParam.getFinanceRole());
//}
//根据备案员名字查询
if(ObjectUtil.isNotEmpty(plOrderParam.getFilingRole())){
queryWrapper.eq("pop4.name",plOrderParam.getFilingRole());
}
//根据创建日期查询
if(ObjectUtil.isNotEmpty(plOrderParam.getCreateDate())){
queryWrapper.apply("date_format(por.create_time,'%Y-%m-%d')>={0}",plOrderParam.getCreateDate
());
}
//根据创建日期（年月）查询
if(ObjectUtil.isNotEmpty(plOrderParam.getCreateYM())){
queryWrapper.like("por.create_time",plOrderParam.getCreateYM());
}
//根据订单类型
if(ObjectUtil.isNotEmpty(plOrderParam.getOrderType())){
queryWrapper.eq("por.order_type",plOrderParam.getOrderType());
}
//业务主管（经理）看到当前区域的所有
if(sysLoginUser.getAdminType()==2&&LoginContext.hasRole("ywbzg")){
for(LongroleId:userRoleIdList){
```

```
SysRolerole=sysRoleService.getById(roleId);
//截取权限 code
String[]roleAddress=role.getCode().split("_");
if(roleAddress.length>1){//代表是辛道的
if(roleAddress[0].equals(SysRoleExceptionEnum.YYB.getMessage())){
//运营部主管看所有地区
}else{
queryWrapper.like("sr.code",roleAddress[0]);
}
//普林的业务部主管看所有(不需要添加)
}
}
}
//业务部部员仅能看到与自己关联的订单
elseif(sysLoginUser.getAdminType()==2&&LoginContext.hasRole("ywbby")){
queryWrapper.eq("pop2.user_id",sysLoginUser.getId());
}
//普通备案人员仅能看到与自己关联的订单
elseif(sysLoginUser.getAdminType()==2&&LoginContext.hasRole("babby")){
queryWrapper.eq("por.record_user",sysLoginUser.getId());
}
//普通下单员仅能看到与自己关联的订单
elseif(sysLoginUser.getAdminType()==2&&LoginContext.hasRole("xdbby")){
queryWrapper.eq("por.create_user",sysLoginUser.getId());
}
}
//5:作废申请审核中，6:已作废这些状态时，前端应该都不显示该订单
//queryWrapper.ne("por.order_status",OrderStatusEnum.APPLICATION_FOR_CANCEL.getCode());
queryWrapper.ne("por.order_status",OrderStatusEnum.CANCELDE.getCode());
//设置逻辑删除插件后，可以删除这行代码
queryWrapper.eq("por.is_deleted",0);
queryWrapper.groupBy("por.id");
//按照更新时间进行降序排序
queryWrapper.orderByDesc("por.update_time");
//分页变成查所有
List<PlOrderVo>plOrderVo=this.baseMapper.page(queryWrapper);
//查询每个订单涉及的样品、期次、收款、分包成本价
List<PlOrderVo>records=plOrderVo;
Page<PlOrderVo>page1=PageFactory.defaultPage();
page1.setRecords(plOrderVo);
page1.setTotal(plOrderVo.size());
for(PlOrderVoorderItem:records){
Stringid=orderItem.getId();
//获取订单对应的样品
```

```
List<PlSampleDto>plSampleList=plOrderSampleService.getSampleListByOrderId(id);
orderItem.setPlSampleList(plSampleList);
//期次（通过订单 id 获取到期次）
List<PlPeriod>periodList=orderPeriodService.getPeriodListByOrderId(id);
orderItem.setPeriodList(periodList);
//收款信息（通过期次 id 获取收款信息）
List<String>periodIds=periodList.stream().filter(item->Objects.equals(item.getPayStatus(),"3"))
.map(PlPeriod::getId).collect(Collectors.toList());
LambdaQueryWrapper<PlCustomer>customerLambdaQueryWrapper=newLambdaQueryWrapper<>();
customerLambdaQueryWrapper.eq(PlCustomer::getId,orderItem.getCustomerId());
booleanhasContract=!StringUtils.isEmpty(plCustomerService.getOne(customerLambdaQueryWrapper)
.getFileId());//是否关联合同
List<OrderPaymentDTO>paymentList=paymentService.getPaymentListByPeriodIds(periodIds,hasContr
act);
//收款信息设置核销员
orderItem.setPaymentList(paymentList);
//设置分包成本价
List<PlOrderSample>sampleList=plOrderSampleService.list(newQueryWrapper<PlOrderSample>().eq(
"order_id",id).eq("is_deleted",0));
List<String>sampleIds=sampleList.stream().map(PlOrderSample::getSampleId).collect(Collectors
.toList());
intsampleSumCost=sampleService.getAmountBySampleIds(sampleIds);
orderItem.setSampleSumCost(sampleSumCost);
}
returnnewPageResult<>(page1);
}
@Override
publicList<PlOrder>list(PlOrderParamplOrderParam){
returnthis.list();
}
@Override
publicList<PlOrder>getByCsId(StringcsId){
LambdaQueryWrapper<PlOrder>plOrderLambdaQueryWrapper=newLambdaQueryWrapper<>();
plOrderLambdaQueryWrapper.eq(PlOrder::getCustomerId,csId);
returnthis.list(plOrderLambdaQueryWrapper);
}
publicvoidcreateOrderNumber(){
}
@Transactional(rollbackFor=Exception.class)
@Override
publicsynchronizedvoidadd(PlOrderParamplOrderParam){
PlOrderplOrder=newPlOrder();
BeanUtil.copyProperties(plOrderParam,plOrder);
```

```
//生成订单编号
IntegerserviceType=plOrderParam.getServiceType();
StringredisKey="order_single_day_counter";
//key 不存在才添加
stringRedisTemplate.opsForValue().setIfAbsent(redisKey,"1");
Stringcounter=stringRedisTemplate.opsForValue().get(redisKey);
counter=String.format("%05d",Integer.parseInt(counter));
SimpleDateFormatdateFormat=newSimpleDateFormat("yyyyMM");
StringnumberStr=dateFormat.format(newDate())+counter;
StringorderNumber=null;
//判断服务类型
switch(serviceType){
case1:
orderNumber="S"+numberStr;
//设置备案状态为未备案
plOrder.setRecordStatus(RecordStatusEnum.UN_RECORD.getCode());
break;
case2:
orderNumber="T"+numberStr;
plOrder.setServiceAmount(null);//检测类型不用设置服务成本价
plOrder.setRecordStatus(null);//检测类型不走审核流程
break;
case3:
orderNumber="ST"+numberStr;
//设置备案状态为未备案
plOrder.setRecordStatus(RecordStatusEnum.UN_RECORD.getCode());
break;
case4:
orderNumber="R"+numberStr;
//设置备案状态为未备案
plOrder.setRecordStatus(RecordStatusEnum.UN_RECORD.getCode());
break;
case5:
orderNumber="E"+numberStr;
//设置备案状态为未备案
plOrder.setRecordStatus(RecordStatusEnum.UN_RECORD.getCode());
break;
case6:
orderNumber="RT"+numberStr;
//设置备案状态为未备案
plOrder.setRecordStatus(RecordStatusEnum.UN_RECORD.getCode());
break;
case7:
orderNumber="ET"+numberStr;
```

```
//设置备案状态为未备案
plOrder.setRecordStatus(RecordStatusEnum.UN_RECORD.getCode());
break;
case8:
orderNumber="RE"+numberStr;
//设置备案状态为未备案
plOrder.setRecordStatus(RecordStatusEnum.UN_RECORD.getCode());
break;
}
plOrder.setOrderNumber(orderNumber);
plOrder.setIsDeleted(0);
//设置订单状态为待确认下单金额
plOrder.setOrderStatus(OrderStatusEnum.ENTER_ORDER_INFO.getCode());
//保存订单
this.save(plOrder);
//添加订单样品关联数据
plOrderSampleService.saveBatchOrderSample(plOrderParam.getPlSampleIds(),plOrder.getId());
//订单编号计数器自增
stringRedisTemplate.opsForValue().increment(redisKey);
//同步 pl_order_personnel 表
//获取当前用户信息
SysLoginUsersysLoginUser=(SysLoginUser)SecurityContextHolder.getContext().getAuthentication(
).getPrincipal();
PlOrderPersonnelplOrderPersonnel=newPlOrderPersonnel();
plOrderPersonnel.setRoleType(RoleTypeEnum.ORDER_ROLE.getType());
plOrderPersonnel.setUserName(sysLoginUser.getName());
plOrderPersonnel.setUserId(sysLoginUser.getId());
plOrderPersonnel.setOrderId(plOrder.getId());
plOrderPersonnel.setIsDeleted(0);
plOrderPersonnelService.save(plOrderPersonnel);
//SysLoginUsersysLoginUser=(SysLoginUser)SecurityContextHolder.getContext().getAuthenticatio
n().getPrincipal();
PlCustomerplCustomer=plCustomerService.getById(plOrder.getCustomerId());
PlOrderPersonnelplOrderPersonnel1=newPlOrderPersonnel();
plOrderPersonnel1.setRoleType(RoleTypeEnum.TRANSACT_ROLE.getType());
plOrderPersonnel1.setUserName(plCustomer.getUserName());
plOrderPersonnel1.setUserId(plCustomer.getSalesman());
plOrderPersonnel1.setOrderId(plOrder.getId());
plOrderPersonnel1.setIsDeleted(0);
plOrderPersonnelService.save(plOrderPersonnel1);
//客户下单后，该客户可以从该业务员所被限制的领取客户人数中去除
relieveCustomerLimit(plCustomer.getSalesman());
//relieveCustomerLimit(plCustomer.getUserName());
//生成订单时，再次修改样品名称（产品名称+样品 X 号）
```

```
//List<PlSampleDto>plSampleDtos=plOrderSampleService.getSampleListByOrderId(plOrder.getId())
;
//for(PlSampleDtoplSampleDto:plSampleDtos){
//PlSampleplSample=newPlSample();
//BeanUtil.copyProperties(plSampleDto,plSample);
//StringsampleName=plSample.getSampleName();
//plSample.setSampleName(plOrderParam.getProductName()+sampleName);
//plSampleService.updateById(plSample);
//}
//生成订单后把样品 X 号设为 0
//如果委托公司是从用户生产公司带出来的（）
if(plOrderParam.getEntrustCompany().equals(plCustomer.getProductCompanyName())){
PlEntrustCompanyplEntrustCompany=newPlEntrustCompany();
plEntrustCompany.setAddress(plCustomer.getProductCompanyAddress());
plEntrustCompany.setPhone(plCustomer.getProductCompanyPhone());
plEntrustCompany.setCompanyName(plCustomer.getCompanyName());
plEntrustCompanyService.save(plEntrustCompany);
//添加用户和委托公司关联数据
PlEntrustCompanyCustomercompanyCustomer=newPlEntrustCompanyCustomer();
companyCustomer.setCustomerId(plOrderParam.getCustomerId());
companyCustomer.setEntrustId(plEntrustCompany.getId());
entrustCompanyCustomerService.save(companyCustomer);
}else{
//添加用户和委托公司关联数据
PlEntrustCompanyCustomercompanyCustomer=newPlEntrustCompanyCustomer();
companyCustomer.setCustomerId(plOrderParam.getCustomerId());
companyCustomer.setEntrustId(plOrderParam.getEntrustCompany());
entrustCompanyCustomerService.save(companyCustomer);
}
//TODO:设置预计出报告时间通知，使用定时器+消息
}
@Transactional(rollbackFor=Exception.class)
@Override
publicvoiddelete(List<PlOrderParam>plOrderParamList){
plOrderParamList.forEach(plOrderParam->{
this.removeById(plOrderParam.getId());
this.removeById(plOrderParam.getUpdateUser());
});
}
@Override
publicPageResult<RecordDto>recordDtoPage(IntegerrecordStatus,StringfinishTime){
SysLoginUsersysLoginUser=LoginContextHolder.me().getSysLoginUser();
List<RecordDto>recordDtos=null;
booleanhqb=LoginContext.hasRole("babby");
```

```
//如果是备案部主管就展示所有
if(sysLoginUser.getAdminType()==1||LoginContext.hasRole("babzg")||LoginContext.hasRole("jsbz
g")||LoginContext.hasRole("zjl")||LoginContext.hasRole("bazy")||LoginContext.hasRole("zjb"))
{
recordDtos=plOrderMapper.recordDtoPage(recordStatus,finishTime);
}elseif(LoginContext.hasRole("babby")||LoginContext.hasRole("jsbbaby")||LoginContext.hasRole
("hqbzj")){
recordDtos=plOrderMapper.recordDtoPage2(recordStatus,finishTime,sysLoginUser.getId());
}
for(RecordDtorecordDto:recordDtos){
SysUsersysUser=sysUserService.getById(recordDto.getBusinessMan());
recordDto.setBusinessMan(sysUser.getName());
List<PlSample>plSampleByOrderId=plSampleMapper.getPlSampleByOrderId(recordDto.getId());
recordDto.setSamples(plSampleByOrderId);
}
Page<RecordDto>page2=PageFactory.defaultPage();
if(recordDtos!=null){
page2.setTotal(recordDtos.size());
}
List<RecordDto>resultList=PageUtil.page(page2,recordDtos);
returnnewPageResult(page2,resultList);
}
@Transactional(rollbackFor=Exception.class)
@Override
publicvoidedit(PlOrderParamplOrderParam){
PlOrderplOrder=this.queryPlOrder(plOrderParam);
BeanUtil.copyProperties(plOrderParam,plOrder);
this.updateById(plOrder);
}
//@Override
//publicPageResult<List<PlOrder>>getOrdersByUserId(StringuserId){
//List<PlOrder>orderList=plOrderMapper.getOrdersByUserId(userId);
//Page<PlOrder>page=PageFactory.defaultPage();
//page.setSize(5);//每页五条
//List<PlOrder>resultList=PageUtil.page(page,orderList);
//returnnewPageResult(page,resultList);
//}
@Override
publicPageResult<List<PlOrder>>getOrdersInUserDetail(PlOrderSimpParamplOrderSimpParam){
List<PlOrder>orderList=plOrderMapper.getOrdersByUserIdAndTime(plOrderSimpParam);
Page<PlOrder>page=PageFactory.defaultPage();
page.setSize(orderList.size());
List<PlOrder>resultList=PageUtil.page(page,orderList);
returnnewPageResult(page,resultList);
```

```
}
@Override
publicPlOrderVodetail(PlOrderParamplOrderParam){
PlOrderVoplOrderVo=baseMapper.getOrderDetail(plOrderParam.getId());
if(plOrderVo==null){
thrownewServiceException(PlOrderExceptionEnum.ID_NOT_EXIST);
}
//获取订单对应的样品 ids
List<PlOrderSample>sampleList=plOrderSampleService.list(newQueryWrapper<PlOrderSample>().eq(
"order_id",plOrderVo.getId()));
List<String>sampleIds=sampleList.stream().map(item->item.getSampleId()).collect(Collectors.t
oList());
//设置分包成本价
intsampleSumCost=sampleService.getAmountBySampleIds(sampleIds);
plOrderVo.setSampleSumCost(sampleSumCost);
//样品信息
List<PlSampleDto>plSampleList=sampleService.getPlSampleListByIds(sampleIds);
plOrderVo.setPlSampleList(plSampleList);
//期次信息
List<PlOrderPeriod>orderPeriodList=orderPeriodService.list(newQueryWrapper<PlOrderPeriod>().
eq("order_id",plOrderVo.getId()).select("period_id"));
List<String>periodIds=orderPeriodList.stream().map(item->item.getPeriodId()).collect(Collect
ors.toList());
List<PlPeriod>periodList=periodService.getPeriodByPeriodId(periodIds);
plOrderVo.setPeriodList(periodList);
//收款信息
List<OrderPaymentDTO>billList=billService.getBillByOrderId(plOrderVo.getId());
plOrderVo.setPaymentList(billList);
returnplOrderVo;
}
/**
*获取订单表
*
*@authorcaizepei
*@date2023-09-1416:59:38
*/
privatePlOrderqueryPlOrder(PlOrderParamplOrderParam){
PlOrderplOrder=this.getById(plOrderParam.getId());
//PlOrderplOrder=this.getById(plOrderParam.getUpdateUser());
if(ObjectUtil.isNull(plOrder)){
thrownewServiceException(PlOrderExceptionEnum.NOT_EXIST);
}
returnplOrder;
}
```

```
//@Override
//publicvoidexport(PlOrderParamplOrderParam){
//List<PlOrder>list=this.list(plOrderParam);
//PoiUtil.exportExcelWithStream("SnowyPlOrder.xls",PlOrder.class,list);
//}
//
@Override
publicStringexport(List<String>ids){
//ids 为空
if(ids.isEmpty()){
thrownewServiceException(PlOrderExceptionEnum.ID_EMPTY_ERROR);
}
//查询出订单列表
List<PlOrderExcel>list=baseMapper.getOrderByIds(ids);
//设置每个订单涉及的样品、期次、收款、分包成本价
for(PlOrderExcelorderItem:list){
Stringid=orderItem.getId();
//获取订单对应的样品
List<PlSampleDto>plSampleList=plOrderSampleService.getSampleListByOrderId(id);
List<PlOrderExcel.PlSampleInfo>plSampleInfos=plSampleList.stream().map(item->{
PlOrderExcel.PlSampleInfoplSampleInfo=newPlOrderExcel.PlSampleInfo();
BeanUtil.copyProperties(item,plSampleInfo);
StringBuilderprojectNames=newStringBuilder();
//将检测项目拼接在一起
for(PlAnalysisProjectproject:item.getProject()){
projectNames.append(project.getProjectName());
projectNames.append("|");
}
projectNames.deleteCharAt(projectNames.length()-1);
plSampleInfo.setProject(projectNames.toString());
returnplSampleInfo;
}).collect(Collectors.toList());
orderItem.setPlSampleList(plSampleInfos);
//期次
List<PlPeriod>periodList=orderPeriodService.getPeriodListByOrderId(id);
List<PlOrderExcel.PlPeriodInfo>plPeriodInfos=periodList.stream().map(item->{
PlOrderExcel.PlPeriodInfoplPeriodInfo=newPlOrderExcel.PlPeriodInfo();
BeanUtil.copyProperties(item,plPeriodInfo);
returnplPeriodInfo;
}).collect(Collectors.toList());
orderItem.setPeriodList(plPeriodInfos);
//收款信息
List<String>periodIds=periodList.stream().filter(item->Objects.equals(item.getPayStatus(),"3
")).map(PlPeriod::getId).collect(Collectors.toList());
```

```
booleanhasContract=StringUtils.isEmpty(orderItem.getContractCode());//是否关联合同
List<OrderPaymentDTO>paymentList=paymentService.getPaymentListByPeriodIds(periodIds,hasContr
act);
//将设置收款方式
paymentList=paymentList.stream().peek(item->{
StringpaymentMethod=item.getPaymentMethod();
if(Objects.equals(paymentMethod,"1")){
item.setPaymentMethod("银行账户");
}elseif(Objects.equals(paymentMethod,"2")){
item.setPaymentMethod("支付宝");
}elseif(Objects.equals(paymentMethod,"3")){
item.setPaymentMethod("微信");
}
}).collect(Collectors.toList());
orderItem.setPaymentList(paymentList);
//设置分包成本价
List<PlOrderSample>sampleList=plOrderSampleService.list(newQueryWrapper<PlOrderSample>().eq(
"order_id",id).eq("is_deleted",0));
List<String>sampleIds=sampleList.stream().map(PlOrderSample::getSampleId).collect(Collectors
.toList());
intsampleSumCost=sampleService.getAmountBySampleIds(sampleIds);
orderItem.setSampleSumCost(sampleSumCost);
//将状态码转化为名字
orderItem.setServiceTypeName(ServiceTypeEnum.getNameByCode(orderItem.getServiceType()));
orderItem.setOrderStatusName(OrderStatusEnum.getNameByCode(orderItem.getOrderStatus()));
orderItem.setRecordStatusName(RecordStatusEnum.getNameByCode(orderItem.getRecordStatus()));
orderItem.setOrderTypeName(OrderTypeEnum.getNameByCode(orderItem.getOrderType()));
Workbookworkbook=ExcelExportUtil.exportExcel(newExportParams(),PlOrderExcel.class,list);
ByteArrayInputStreamin=null;
ByteArrayOutputStreamos=null;
FileDTOfileDTO=null;
try{
os=newByteArrayOutputStream();
workbook.write(os);
byte[]b=os.toByteArray();
in=newByteArrayInputStream(b);
os.close();
fileDTO=FileUtil.uploadFile(in,false,FileConstantEnum.TEMP.getScene(),"SnowyPlOrder.xls");
}catch(Exceptione){
e.printStackTrace();
}finally{
if(in!=null){
try{
in.close();
```

```
}catch(IOExceptione){
e.printStackTrace();
}
}
if(os!=null){
try{
os.close();
}catch(IOExceptione){
e.printStackTrace();
}
}
}
returnfileDTO.getUrl();
}
/**
*下单员申请订单作废业务实现
*
*@paramcancelOrderDTO
*/
@Override
publicvoidcancelOrder(ApplyOperateDTOcancelOrderDTO){
Stringreason=cancelOrderDTO.getReason();
StringorderNumber=cancelOrderDTO.getOrderNumber();
//判断申请原因是否为空，空则抛出异常
if(null==reason||reason.trim().isEmpty()){
thrownewServiceException(PlOrderExceptionEnum.REASON_NOT_EXIST);
}
//一.修改订单表
//1.保存当前订单状态,通过订单编号查询
QueryWrapper<PlOrder>queryWrapper2=newQueryWrapper<>();
queryWrapper2.lambda().eq(PlOrder::getOrderNumber,orderNumber);
PlOrderorder=this.getOne(queryWrapper2);
//保存修改前的订单状态
intorderStatus=order.getOrderStatus();
//判断订单是否处于“待确认下单金额”或“未收款”，不处于则不能“申请订单作废”
if(orderStatus!=1&&orderStatus!=2){
thrownewServiceException(PlOrderExceptionEnum.ORDER_STATUS_NOT_MEET);
}
//2.修改订单状态
order.setOrderStatus(OrderStatusEnum.APPLICATION_FOR_CANCEL.getCode());
order.setServiceType(2);
this.updateById(order);
//二.填写消息通知表 sys_notice 和审批表
SysLoginUsersysLoginUser=LoginContextHolder.me().getSysLoginUser();
```

```
SysNoticeParamsysNoticeParam=newSysNoticeParam();
sysNoticeParam.setContractId(order.getContractId());
sysNoticeParam.setOrderId(order.getId());
sysNoticeParam.setType(NoticeTypeEnum.ORDER_CANCELLATION.getCode());
Stringtemplate=NoticeTypeEnum.ORDER_CANCELLATION.getTemplate().replace("${xdy}",sysLoginUser
.getName());
sysNoticeParam.setTitle(template);
sysNoticeParam.setApproveReason(reason);
sysNoticeParam.setContent(reason);
sysNoticeParam.setOrderStatus(orderStatus);
plNoticeService.add(sysNoticeParam);
}
/**
*申请期次变更
*
*@paramchangePeriodDTO
*/
@Override
publicvoidchangePeriod(ApplyOperateDTOchangePeriodDTO){
Stringreason=changePeriodDTO.getReason();
StringorderNumber=changePeriodDTO.getOrderNumber();
//判断申请原因是否为空，空则抛出异常
if(null==reason||reason.trim().isEmpty()){
thrownewServiceException(PlOrderExceptionEnum.REASON_NOT_EXIST);
}
//判断用户输入的”处理人员“是否存在
//Stringname=changePeriodDTO.getName();
//SysUsersysUser=sysUserService.getSysUserByName(name);
//if(BeanUtil.isEmpty(sysUser)){
//thrownewServiceException(PlOrderExceptionEnum.USER_NOT_EXIST);
//}
//StringpromoterId=String.valueOf(sysUser.getId());
//一.修改订单表
//1.保存当前订单状态,通过订单编号查询
QueryWrapper<PlOrder>queryWrapper2=newQueryWrapper<>();
queryWrapper2.lambda().eq(PlOrder::getOrderNumber,orderNumber);
PlOrderorder=this.getOne(queryWrapper2);
intorderStatus=order.getOrderStatus();
//判断订单是否处于“未收款”，不处于则不能“变更期次”
if(orderStatus!=2){
thrownewServiceException(PlOrderExceptionEnum.ORDER_STATUS_NOT_MEET);
}
//2.修改订单状态
order.setServiceType(2);
```

```
order.setOrderStatus(7);
this.updateById(order);
//获取当前用户信息
SysLoginUsersysLoginUser=LoginContextHolder.me().getSysLoginUser();
SysNoticeParamsysNoticeParam=newSysNoticeParam();
sysNoticeParam.setContractId(order.getContractId());
sysNoticeParam.setOrderId(order.getId());
sysNoticeParam.setType(NoticeTypeEnum.CHANGE_PERIOD.getCode());
Stringtemplate=NoticeTypeEnum.CHANGE_PERIOD.getTemplate().replace("${ywy}",sysLoginUser.getN
ame());
sysNoticeParam.setTitle(template);
sysNoticeParam.setApproveReason(reason);
sysNoticeParam.setContent(reason);
sysNoticeParam.setOrderStatus(orderStatus);
plNoticeService.add(sysNoticeParam);
}
//获取订单详情
@Override
publicPlOrdergetOrderById(StringorderId){
PlOrderplOrder=this.getById(orderId);
returnplOrder;
}
@Override
publicPlOrdersetOrderRecorder2(OrderRecorderVoorderRecorderVo){
PlOrderplOrder=newPlOrder();
plOrder.setId(orderRecorderVo.getOrderId());
plOrder.setRecordUser(Long.parseLong(orderRecorderVo.getRecorderId2()));
PlOrderorder=this.getById(plOrder);
baseMapper.updateById(plOrder);
PlOrderPersonnelplOrderPersonnel=newPlOrderPersonnel();
plOrderPersonnel.setRoleType(RoleTypeEnum.FILING_ROLE.getType());
plOrderPersonnel.setUserName(sysUserService.getNameByUserId(Long.valueOf(orderRecorderVo.get
RecorderId2())));
plOrderPersonnel.setUserId(Long.valueOf(orderRecorderVo.getRecorderId2()));
plOrderPersonnel.setOrderId(orderRecorderVo.getOrderId());
plOrderPersonnel.setIsDeleted(0);
plOrderPersonnelService.save(plOrderPersonnel);
returnorder;
}
@Override
@Transactional(rollbackFor=Exception.class)
publicvoidimportOrder(MultipartFilefile){
//获取到 workbook 对象
Workbookworkbook=PoiExcelUtils.creatWorkBook(file);
```

```
//循环获取到多个 sheet 表
for(intsheetNum=0;sheetNum<workbook.getNumberOfSheets();sheetNum++){
Sheetsheet=workbook.getSheetAt(sheetNum);//获取表格
//校验 sheet 是否合法
if(sheet==null){
continue;
}
RowfirstRow=sheet.getRow(sheet.getFirstRowNum());//获取第一行，一般是标题
introwStart=1;//第一行开始遍历,数据起始行
introwEnd=sheet.getLastRowNum();//获取有记录的行数，即：最后有数据的行是第 n 行，前面有 m 行是空行没数据，则返回 n-m
for(introwNum=rowStart;rowNum<=rowEnd;rowNum++){//行
Rowrow=sheet.getRow(rowNum);
log.info("第{}行",row.getRowNum());
if(null==row){
continue;
}
//每一行
//一个订单
PlOrderplOrder=newPlOrder();
//多个检测项目
List<PlAnalysisProject>plAnalysisProjects=newArrayList<>();
//一个检测公司
PlAnalysisCompanyplAnalysisCompany=newPlAnalysisCompany();
//一个样品
PlSampleplSample=newPlSample();
//一个客户
PlCustomerplCustomer=newPlCustomer();
//一个业务员
SysUsersysUser=newSysUser();
//一个下单员
PlOrderPersonnelplOrderPersonnel=newPlOrderPersonnel();
//判断是否公海
booleanqueryIsPublic=false;
//每一行的业务员
SysUsersysUser2=newSysUser();
for(intcellNum=0;cellNum<9;cellNum++){//列
Cellcell=row.getCell(cellNum);
if(cell!=null){
Stringcontent=PoiExcelUtils.convertCellValueToString(cell);
log.info("第"+row.getRowNum()+"行，第"+cellNum+"列，内容："+content);
switch(cellNum){
case1://业务员(由于业务员当时没有关联到 pop 表，是通过客户表的业务员 id 获取)
if(content==null){
```

```
queryIsPublic=true;
}
else{
//判断业务员是否已存在
SysUsersysUserByName=sysUserService.getSysUserByName(content);
BeanUtil.copyProperties(sysUserByName,sysUser2);
//业务员不存在，新增业务员
if(sysUser2.getId()==null){
sysUser.setAccount("123456");
sysUser.setPwdHashValue("123456");
sysUser.setName(content);
sysUser.setSex(1);
sysUser.setAdminType(0);
sysUser.setStatus(1);//旧数据的用户（账号冻结）
sysUserService.save(sysUser);
queryIsPublic=false;
}else{
queryIsPublic=false;
}
}
break;
case2:
if(content==null){
thrownewServiceException(4000,"客户名称不能为空");
}
//if(plCustomerService.getCustomersByName(content).size()>0){
PlCustomerbyCompanyName=plCustomerService.getByCompanyName(content);
if(byCompanyName!=null){
log.info("已有客户[{}]",content);
plCustomer=byCompanyName;
break;
}
log.info("新增客户[{}]",content);
//客户名称
plCustomer.setCompanyName(content);
plCustomer.setUserName("无");
if(queryIsPublic){
plCustomer.setIsPublicUser(1);
}
//客户表中关联业务员
if(sysUser.getId()!=null){
plCustomer.setSalesman(sysUser.getId());
}else{
plCustomer.setSalesman(sysUser2.getId());
```

```
}
plCustomerService.save(plCustomer);
break;
case3://服务类型
if(content==null||content.equals("")){
plOrder.setServiceType(null);
}elseif(content.equals("服务")){
plOrder.setServiceType(1);
}elseif(content.equals("检测")){
plOrder.setServiceType(2);
}elseif(content.equals("服务+检测")||content.equals("检测+服务")){
plOrder.setServiceType(3);
}elseif(content.equals("备案")){
plOrder.setServiceType(4);
}elseif(content.equals("安评")){
plOrder.setServiceType(5);
}elseif(content.equals("备案+检测")||content.equals("检测+备案")){
plOrder.setServiceType(6);
}elseif(content.equals("安评+检测")||content.equals("检测+安评")){
plOrder.setServiceType(7);
}elseif(content.equals("备案+安评")||content.equals("安评+备案")){
plOrder.setServiceType(8);
}else{
thrownewServiceException(4000,"无该服务类型："+content);
}
break;
case4://送检时间
shortformat=cell.getCellStyle().getDataFormat();
if(content==null||content.equals("")){
thrownewServiceException(4000,"送检时间不能为空");
}
if(format==14||format==178){
try{
doublevalue=cell.getNumericCellValue();
Datedate=DateUtil.getJavaDate(value);
ZoneIdzoneId=ZoneId.systemDefault();
Instantinstant=date.toInstant();
LocalDatelocalDate=instant.atZone(zoneId).toLocalDate();
plSample.setInspectionTime(localDate);
}catch(Exceptione){
System.out.println(e.getMessage());
thrownewServiceException(4000,"送检时间不能为空或送检时间格式错误,时间格式为（2023-12-13）");
}
}else{
```

```
thrownewServiceException(4000,"送检时间不能为空或送检时间格式错误,时间格式为（2023-12-13）");
}
break;
case5://检测公司
if(content==null){
thrownewServiceException(4000,"检测公司不能为空");
}
PlAnalysisCompanybyName=plAnalysisCompanyService.getByName(content);
if(byName!=null){
plAnalysisCompany=byName;
log.info("已存在分包单位： {}",byName.getCompanyName());
break;
}
log.info("新建分包单位： {}",content);
plAnalysisCompany.setCompanyName(content);
plAnalysisCompany.setCompanyAbbreviation("无");
plAnalysisCompany.setDirector("无");
plAnalysisCompanyService.save(plAnalysisCompany);
break;
case6://样品名称
if(content==null){
thrownewServiceException(4000,"样品名称不能为空");
}
plSample.setSampleName(content);
plSample.setCompanyId(plAnalysisCompany.getId());
plSample.setSampleCode("");
plSample.setProductNumber("无");
plSampleService.save(plSample);
break;
case7://检测项目
if(content==null){
thrownewServiceException(4000,"检测项目不能为空");
}
String[]projects=content.split("\\+");
//for(StringprojectName:projects){
//if(plAnalysisCompanyService.hasAnalysisProjectById(plAnalysisCompany.getId(),projectName))
{
//log.info("分包单位[{}],已有检测项目[{}]",plAnalysisCompany.getCompanyName(),projectName);
//content=content.replace("projectName","");
//}
//}
//if(content.equals("+"))
//
for(Stringproject:projects){
```

```
if(plAnalysisCompanyService.hasAnalysisProjectById(plAnalysisCompany.getId(),project)){
log.info("分包单位[{}],已有检测项目[{}]",plAnalysisCompany.getCompanyName(),project);
continue;
}
log.info("分包单位[{}],新增检测项目[{}]",plAnalysisCompany.getCompanyName(),content);
PlAnalysisProjectplAnalysisProject=newPlAnalysisProject();
plAnalysisProject.setProjectName(project);
plAnalysisProject.setAmount(0);
plAnalysisProjectService.save(plAnalysisProject);
//关联样品
PlSampleProjectplSampleProject=newPlSampleProject();
plSampleProject.setSampleId(plSample.getId());
plSampleProject.setProjectId(plAnalysisProject.getId());
plSampleProjectService.save(plSampleProject);
//检测项目与检测公司关联
PlCompanyProjectplCompanyProject=newPlCompanyProject();
plCompanyProject.setProjectId(plAnalysisProject.getId());
plCompanyProject.setCompanyId(plAnalysisCompany.getId());
plCompanyProjectService.save(plCompanyProject);
}
break;
case8://下单员
plOrderPersonnel.setRoleType(1);
plOrderPersonnel.setUserName(content);
break;
default:
break;
}
}
}
//新增订单
plOrder.setOrderNumber("无");
plOrder.setCustomerId(plCustomer.getId());
plOrder.setOrderStatus(4);
plOrder.setReturnAmount(0);
this.save(plOrder);
//下单人
plOrderPersonnel.setOrderId(plOrder.getId());
plOrderPersonnelService.save(plOrderPersonnel);
//订单与样品关联
PlOrderSampleplOrderSample=newPlOrderSample();
plOrderSample.setOrderId(plOrder.getId());
plOrderSample.setSampleId(plSample.getId());
plOrderSampleService.save(plOrderSample);
```

```
}
}
}
/**
*为订单录入下单金额
*
*@paramorderReceivableVo
*/
@Transactional(rollbackFor=Exception.class)
@Override
publicvoidsetReceivable(OrderReceivableVoorderReceivableVo){
//订单状态必须是处于“待确认下单金额”状态
StringorderId=orderReceivableVo.getOrderId();
PlOrderplOrder=this.getById(orderId);
if(plOrder==null){
thrownewServiceException(PlOrderExceptionEnum.STATUS_ERORR);
}
if(!OrderStatusEnum.ENTER_ORDER_INFO.getCode().equals(plOrder.getOrderStatus())){
thrownewServiceException(PlOrderExceptionEnum.STATUS_ERORR);
}
////根据当前用户的组织机构 id，判断是否具有操作权限
////获取当前登录用户的组织机构 id
//SysOrgorg=sysOrgService.getById(loginContext.getSysLoginUserOrgId());
//if(!org.getName().equals("业务部")){
////抛出准确异常
//thrownewServiceException(PlOrderExceptionEnum.USER_NOT_POEER);
//}
//先根据订单 id 查询对应的订单样品表的数据
QueryWrapper<PlOrderSample>queryWrapperPlSample=newQueryWrapper<>();
queryWrapperPlSample.lambda().eq(PlOrderSample::getOrderId,orderId);
List<PlOrderSample>plOrderSamples=plOrderSampleService.list(queryWrapperPlSample);
//总分包成本价
intamount=0;
if(plOrderSamples!=null){
//拿到订单对应的各个样品 id
List<String>plSampleIds=newArrayList<>();
for(PlOrderSampleplOrderSample:plOrderSamples){
plSampleIds.add(plOrderSample.getSampleId());
}
//根据样品 id 列表获取总分包成本价
amount=plSampleService.getOrderAmount(plSampleIds);
}
//根据订单 id 获取服务成本价
intserviceAmount=0;
```

```
if(plOrder.getServiceAmount()!=null){
serviceAmount=plOrder.getServiceAmount();
}
//累加每期次的预计收款金额算出订单的成交金额
intorderAmount=0;
//判断录入的期次信息是否存在
if(orderReceivableVo.getPlOrderPeriodVoList()==null){
thrownewServiceException(PlOrderExceptionEnum.PERIOD_ERROR);
}
for(PlOrderPeriodVoplOrderPeriodVo:orderReceivableVo.getPlOrderPeriodVoList()){
//累加每期次的预计收款金额
orderAmount+=plOrderPeriodVo.getExpectIncomeMoney();
}
//判断成交金额是否小于总分包成本价*成本价利率+服务成本价
if((amount*Float.parseFloat(plOrder.getRate()))+serviceAmount>orderAmount){
thrownewServiceException(PlOrderExceptionEnum.DATA_INPUT_ERORR);
}
plOrder.setReceivable(orderAmount);
baseMapper.updateById(plOrder);
//设置期次表实体类信息
intsize=1;
for(PlOrderPeriodVoplOrderPeriodVo:orderReceivableVo.getPlOrderPeriodVoList()){
//判断录入的期次信息是否存在出现空值或金额小于或等于 0 的情况
if(plOrderPeriodVo.getExpectIncomeTime()==null||plOrderPeriodVo.getExpectIncomeMoney()==null
||plOrderPeriodVo.getExpectIncomeMoney()<=0){
thrownewServiceException(PlOrderExceptionEnum.EXPECT_PERIOD_NULL);
}
PlPeriodplPeriod=newPlPeriod();
plPeriod.setExpectIncomeTime(plOrderPeriodVo.getExpectIncomeTime());
plPeriod.setExpectIncomeMoney(plOrderPeriodVo.getExpectIncomeMoney());
plPeriod.setPayStatus("1");
plPeriod.setCompanyName(plCustomerService.getById(plOrder.getCustomerId()).getCompanyName())
;
plPeriod.setCurrPeriod(size+"/"+orderReceivableVo.getPlOrderPeriodVoList().size());
plPeriod.setIsDeleted(0);
plPeriod.setCollection(0);
plPeriod.setCreateTime(newDate());
plPeriod.setCreateUser(String.valueOf(loginContext.getSysLoginUserId().longValue()));
//在期次表中插入记录
plPeriodService.add(plPeriod);
//在订单期次映射表中插入记录
PlOrderPeriodplOrderPeriod=newPlOrderPeriod();
plOrderPeriod.setOrderId(orderId);
plOrderPeriod.setPeriodId(plPeriod.getId());
```

```
plOrderPeriod.setIsDeleted(0);
plOrderPeriodService.save(plOrderPeriod);
size++;
}
//同步 pl_order_personnel 表
//获取当前用户信息
//SysLoginUsersysLoginUser=(SysLoginUser)SecurityContextHolder.getContext().getAuthenticatio
n().getPrincipal();
//PlOrderPersonnelplOrderPersonnel=newPlOrderPersonnel();
//plOrderPersonnel.setRoleType(RoleTypeEnum.TRANSACT_ROLE.getType());
//plOrderPersonnel.setUserName(sysLoginUser.getName());
//plOrderPersonnel.setUserId(sysLoginUser.getId());
//plOrderPersonnel.setOrderId(plOrder.getId());
//plOrderPersonnel.setIsDeleted(0);
//plOrderPersonnelService.save(plOrderPersonnel);
UpdateWrapper<PlOrder>updateWrapper=newUpdateWrapper<>();
updateWrapper.eq("id",orderReceivableVo.getOrderId());
//设置成交金额
updateWrapper.set("amount",orderAmount);
//订单状态变为“已确认下单金额”
updateWrapper.set("order_status",OrderStatusEnum.AMOUNT_TO_BE_CONFIRMED.getCode());
//订单的应收金额变为成交金额
updateWrapper.set("receivable",orderAmount);
this.update(updateWrapper);
}
/**
*查询期次表数据
*
*@paramorderId
*@paramexpectIncomeTime
*/
@Transactional(rollbackFor=Exception.class)
@Override
publicIntegergetPeriodExpectIncomeTime(StringorderId,StringexpectIncomeTime){
////根据当前用户的组织机构 id，判断是否具有操作权限
////获取当前登录用户的组织机构 id
//SysOrgorg=sysOrgService.getById(loginContext.getSysLoginUserOrgId());
//if(!org.getName().equals("财务部")){
////抛出准确异常
//thrownewServiceException(PlOrderExceptionEnum.USER_NOT_POEER);
//}
//先根据订单 id 查询对应的订单期次表的数据
QueryWrapper<PlOrderPeriod>queryWrapperOrderPeriod=newQueryWrapper<>();
queryWrapperOrderPeriod.select("id","order_id","period_id").eq("order_id",orderId);
```

```
List<PlOrderPeriod>plOrderPeriodList=plOrderPeriodService.list(queryWrapperOrderPeriod);
if(plOrderPeriodList!=null){
//根据期次 id 查询对应的期次表的数据
for(PlOrderPeriodplOrderPeriod:plOrderPeriodList){
PlPeriodplPeriod=plPeriodService.selectOne(plOrderPeriod.getPeriodId());
if(plPeriod.getExpectIncomeTime().toString().equals(expectIncomeTime)){
returnplPeriod.getExpectIncomeMoney();
}
}
}
@Override
publicPageResult<PlPayment>page(PlPaymentParamplPaymentParam){
QueryWrapper<PlPayment>queryWrapper=newQueryWrapper<>();
if(ObjectUtil.isNotNull(plPaymentParam)){
//根据期次 id 查询
if(ObjectUtil.isNotEmpty(plPaymentParam.getPeriodId())){
queryWrapper.lambda().eq(PlPayment::getPeriodId,plPaymentParam.getPeriodId());
}
//根据收款金额（单位为分）查询
if(ObjectUtil.isNotEmpty(plPaymentParam.getAmount())){
queryWrapper.lambda().eq(PlPayment::getAmount,plPaymentParam.getAmount());
}
//根据发票编号查询
if(ObjectUtil.isNotEmpty(plPaymentParam.getInvoiceCode())){
queryWrapper.lambda().eq(PlPayment::getInvoiceCode,plPaymentParam.getInvoiceCode());
}
//根据收款账号查询
if(ObjectUtil.isNotEmpty(plPaymentParam.getPaymentAccount())){
queryWrapper.lambda().eq(PlPayment::getPaymentAccount,plPaymentParam.getPaymentAccount());
}
//根据收款方式（银行账号，支付宝，微信）查询
if(ObjectUtil.isNotEmpty(plPaymentParam.getPaymentMethod())){
queryWrapper.lambda().eq(PlPayment::getPaymentMethod,plPaymentParam.getPaymentMethod());
}
}
returnnewPageResult<>(this.page(PageFactory.defaultPage(),queryWrapper));
}
@Override
publicList<PlPayment>list(PlPaymentParamplPaymentParam){
returnthis.list();
}
@Override
publicvoidadd(PlPaymentParamplPaymentParam){
PlPaymentplPayment=newPlPayment();
```

```
BeanUtil.copyProperties(plPaymentParam,plPayment);
this.save(plPayment);
}
@Transactional(rollbackFor=Exception.class)
@Override
publicvoiddelete(List<PlPaymentParam>plPaymentParamList){
plPaymentParamList.forEach(plPaymentParam->{
this.removeById(plPaymentParam.getId());
});
}
@Transactional(rollbackFor=Exception.class)
@Override
publicvoidedit(PlPaymentParamplPaymentParam){
PlPaymentplPayment=this.queryPlPayment(plPaymentParam);
BeanUtil.copyProperties(plPaymentParam,plPayment);
this.updateById(plPayment);
}
@Override
publicPlPaymentdetail(PlPaymentParamplPaymentParam){
returnthis.queryPlPayment(plPaymentParam);
}
/**
*获取收款表
*
*@authorcaizepei
*@date2023-09-1417:01:09
*/
privatePlPaymentqueryPlPayment(PlPaymentParamplPaymentParam){
PlPaymentplPayment=this.getById(plPaymentParam.getId());
if(ObjectUtil.isNull(plPayment)){
thrownewServiceException(PlPaymentExceptionEnum.NOT_EXIST);
}
returnplPayment;
}
@Override
publicvoidexport(PlPaymentParamplPaymentParam){
List<PlPayment>list=this.list(plPaymentParam);
PoiUtil.exportExcelWithStream("SnowyPlPayment.xls",PlPayment.class,list);
}
@Override
publicList<OrderPaymentDTO>getPaymentListByPeriodIds(List<String>periodIds,booleanhasContrac
t){
//为空直接返回
if(periodIds==null||periodIds.isEmpty()){
```

```
returnnull;
}
//如果是没有关联合同的，则直接去收款表查
if(!hasContract){
returnbaseMapper.getPaymentInfoByPeriodIds(periodIds);
}else{
//如果是关联合同的，则有可能一部分在账单表，一部分在收款表
List<OrderPaymentDTO>res=newArrayList<>();
for(StringperiodId:periodIds){
OrderPaymentDTOpaymentDTO=baseMapper.getPaymentInfoByPeriodId(periodId);
if(paymentDTO==null){
//不是通过录入收款金额的方式,去账单表里找,此时期次的实际收款金额等于应收金额
paymentDTO=billService.getBillInfoByPeriodId(periodId);
}
res.add(paymentDTO);
}
returnres;
}
}
@Override
    @Transactional
    public BillExportVO export(PeriodExportParam periodExportParam) {
//        //        根据期次 id 找到订单 id
//        for (String periodId : periodExportParam.getPeriodIdList()) {
//                                                    PlOrderPeriod    orderByPeriodId    =
plOrderPeriodService.getOrderByPeriodId(periodId);
//
//            // 同步 pl_order_personnel 表
//            // 获取当前用户信息
//                                          SysLoginUser    sysLoginUser    =    (SysLoginUser)
SecurityContextHolder.getContext().getAuthentication().getPrincipal();
//            PlOrderPersonnel plOrderPersonnel = new PlOrderPersonnel();
//            plOrderPersonnel.setRoleType(6);
//            plOrderPersonnel.setUserName(sysLoginUser.getName());
//            plOrderPersonnel.setUserId(sysLoginUser.getId());
//            plOrderPersonnel.setOrderId(orderByPeriodId.getOrderId());
//            plOrderPersonnel.setPeriodId(periodId);
//            plOrderPersonnel.setIsDeleted(0);
//            plOrderPersonnelService.save(plOrderPersonnel);
//        }
        // 本期结算
        List<PlPeriodVO>                                curPeriods                                =
plPeriodMapper.queryByIdList(periodExportParam.getPeriodIdList());
        // 上期未付款
```

```
        List<PlPeriodVO>                                lastPeriods                                =
plPeriodMapper.queryPreviousUnpaid(periodExportParam.getCompany());
        if (curPeriods.size() + lastPeriods.size() <= 0) {
            throw new ServiceException(CommonExceptionEnum.NO_PERIOD);
        }
//          生成订单：期次状态为结算中
        for (PlPeriodVO periodVO : curPeriods) {
            PlPeriod plPeriod = new PlPeriod();
            plPeriod.setId(periodVO.getId());
            plPeriod.setPayStatus("2");
            this.updateById(plPeriod);
        }
        for (PlPeriodVO periodVO : lastPeriods) {
            PlPeriod plPeriod = new PlPeriod();
            plPeriod.setId(periodVO.getId());
            plPeriod.setPayStatus("2");
            this.updateById(plPeriod);
        }
        return generateDoc(curPeriods, lastPeriods, periodExportParam.getCompany());
    }
    /**
     * 根据 id 修改付款状态
     *
     * @param idList
```